University of Alberta

SYNTHESIS OF RING-BASED RESTORABLE NETWORKS

by

JAMES BRENT SLEVINSKY

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Electrical and Computer Engineering

Edmonton, Alberta
Spring 1999

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-40113-8

Canada

UNIVERSITY OF ALBERTA

Library Release Form

Name of Author: James Brent Slevinsky

Title of Thesis: Synthesis of Ring-Based Restorable Networks

Degree: Master of Science

Year this Degree Granted: 1999

Permission is hereby granted to the University of Alberta to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly, or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

James Brent Slevinsky

James Brent Slevinsky, P. Eng.
413 Heffernan Drive
Edmonton, Alberta
T6R 1W4

Date: 04 January 1999

Quotations

"Excuse ME, Professor Brainiac, but I WORKED in a nuclear plant for ten years-- and, uh, I think I know how a PROTON ACCELERATOR works!"

--Homer Simpson, before the accident

"Just like the universe, as I get older I expand and get cooler"

--unknown (from the Internet)

I PUT TOGETHER A TIME LINE FOR YOUR PROJECT
I STARTED BY REASONING THAT ANYTHING I DON'T UNDERSTAND IS EASY TO DO
PHASE ONE: DESIGN A CLIENT-SERVER ARCHITECTURE FOR OUR WORLD-WIDE OPERATIONS
TIME: SIX MINUTES

DILBERT reprinted by permission of United Feature Syndicate, Inc.

UNIVERSITY OF ALBERTA

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **SYNTHESIS OF RING-BASED RESTORABLE NETWORKS** submitted by **James Brent Slevinsky** in partial fulfillment of the requirements for the degree of **Master of Science**.

Wayne D. Grover (Supervisor)

Michael H. MacGregor (Co-Supervisor)

Arup Basu (External)

Bruce Cockburn (Internal)

Date: Dec 22/98

Dedicated to my wife Anna, my two sons, John and Luke, and to the memory of my grandmother, Jennie Malanchuk, a pioneer who through her perseverance in life provided the ultimate role model.

Abstract

Societal requirements for highly reliable broadband communications are growing. The key to providing these information conduits is a survivable telecommunications transport infrastructure, of which self-healing ring systems are the present building block of choice. The design of networks utilizing multiple interconnected self-healing rings is extremely complex. Although planners have analysis tools that can assist with the design of such networks, these tools only evaluate the designs presented to them. An automated synthesis system is required in order to facilitate more efficient network designs. This thesis describes the key facets of ring network design, outlines existing approaches to the problem, and introduces and characterizes RingBuilder™, a new system for self-healing ring design synthesis[1]. RingBuilder™ is an iterative heuristic-based framework which generates near-optimal multiple ring designs for large, real-world networks. In this thesis, RingBuilder™ is tested extensively on metropolitan and long-haul networks, revealing several significant design principles for ring network synthesis.

1. RingBuilder™ is a trademark of TR*Labs*.

Acknowledgments

I would like to thank my wife Anna for her extremely strong support during my term at TRLabs and subsequently during my time as a graduate student. This thesis and the work it describes would not have been done without her limitless love, support, and understanding. I would also like to thank my parents, Alexander and Bessie Slevinsky, and my parents-in-law, Jerry and Roma Prystajecky for their assistance and support during this journey. They were always there when assistance was required.

I would like to thank my supervisors, Dr. Wayne D. Grover and Dr. Michael H. MacGregor for their technical support and guidance during this 6 week project that turned into a 6 year journey resulting in RingBuilder™.

I would like to thank all of the staff and students at TRLabs, for providing such an exhilarating environment in which to do research and have fun. In particular, two TRLabs staff members, Dr. Michael Leung and Mr. Wayne Olsen, were always available for technical and non-technical discussions, and I thank them for that. Members of the Network Systems Mafia, that eclectic collection of current and former students, as well as current and former staff, made the work truly enjoyable. Mr. Demetrios Stamatelakis, a charter member of the NS Mafia, always willingly provided assistance when unexpected challenges arose.

Dr. Roger Pederson, Mr. Peng Tan, Mr. Craig Dobson, and Mr. Kirk Mahon, my managers at TELUS Communications during my involvement with TRLabs have been extremely supportive of my work. My colleagues at TELUS, including Mr. Jim Borynec, Mr. Allan Dakin, Mr. Craig Miller, Mr. Dave Moore and Mr. Jason Palm all helped out with the juggling act of *Jim the Student* and *Jim the TELUSian*.

Table of Contents

1. Introduction 1
   1.1 Design of Transport Networks 1
   1.2 Design Problem Complexity 3
   1.3 Research Objectives 6
   1.4 Thesis Organization 6

2. Types of Self-Healing Rings 7
   2.1 Introduction 7
   2.2 Relevant Ring Concepts and Terminology 7
   2.3 Types of SHRs 10
      2.3.1 Bidirectional Line Switched Rings (BLSRs) 11
      2.3.2 Unidirectional Path Switched Rings (UPSRs) 14
      2.3.3 Unidirectional Line Switched Ring 15
      2.3.4 Bidirectional path switched ring 16
   2.4 Ring Loading 18
      2.4.1 BLSR System Capacity 18
      2.4.2 UPSR System Capacity 20
      2.4.3 Add-Drop Multiplexer Considerations 21
      2.4.4 Summary 22
   2.5 Conclusion 23

3. Networks of Multiple SHRs: Design Principles and Issues 24
   3.1 Network Topology Considerations 24
      3.1.1 Network Connectivity 24
      3.1.2 Node and Span Constraints 26
         3.1.2.1 Node Constraints 26
         3.1.2.2 Span Constraints 27
   3.2 Working Path Placement Considerations 27
      3.2.1 Demand Bundling 27
      3.2.2 Demand Routing 28
   3.3 System Technology Considerations 30
      3.3.1 Modularity 31
      3.3.2 ADM Constraints 31
         3.3.2.1 ADM Port Count Limits 31
         3.3.2.2 Active ADM Count 31
   3.4 Design Considerations 32
      3.4.1 Design Technology 33
      3.4.2 Optimization Metrics 34
         3.4.2.1 Redundancy 35
         3.4.2.2 Balance 37
         3.4.2.3 Capture 39

3.4.2.4 Ring Cost 42
3.4.3 Ring Topology Selection 43
3.4.4 Active Node Optimization 43
3.4.5 Demand Route Loading 44
3.5 Existing Approaches to the Design Problem 44
3.5.1 Nortel SONET Planner 45
3.5.2 Genetic Algorithms 47
3.5.3 Bellcore SONET Toolkit 48
3.5.4 Hierarchical Rings 49
3.5.5 Eulerian Ring Covers 50
3.6 Conclusion 50

4. Synthesis of SHR Networks with RingBuilder™ 51
4.1 Introduction 51
4.2 Basic RingBuilder™ Framework 52
4.3 Cycle Finding 54
4.4 Demand Routing 55
4.5 Ring Candidate Loading 56
4.5.1 Demand Route Sorting Options 56
4.5.2 Demand Route Loading Options 57
4.5.3 Active Node/ Glass Through Node Optimization 58
4.6 Choosing Rings 58
4.6.1 Balance vs. Capture Optimization 59
4.6.2 Direct Cost Optimization via Specific Progress 60
4.7 RingBuilder™ Detailed Description 62
4.7.1 Single Technology Balance vs. Capture Optimization 62
4.7.2 Multiple Technology/Direct Cost Optimization 64
4.8 Design Example 67
4.9 Conclusion 70

5. Experimental Results 72
5.1 Test Networks 72
5.2 Cost Models 79
5.3 Experimental Test Cases 80
5.3.1 Design Evaluation Criteria 83
5.3.2 Results and Discussion 83
5.3.3 Correlation Between Cost and Measured Parameters 124
5.4 Conclusions 139

6. Concluding Discussion 141
6.1 Overall results 141
6.1.1 Software Development Observations 141
6.1.2 General Findings About Ring Network Designs 143
6.1.3 Design Synthesis Observations 144

6.2 Uses and Adaptations for Present Work 145
6.3 Future Research 145
6.3.1 Transition Design Capability 145
6.3.2 Improved Cost Models 146
6.3.3 Mixed Mode Design 146
6.3.4 WDM Network Design 146
6.4 Conclusion 147

References 148

Appendix A. RingBuilder™ Implementation Testing 150

A.1 Test Aim 150
A.2 Testing Methodology 150
A.2.1 Module Testing 150
A.2.2 Memory Leak Testing 151
A.2.3 Manual Verification 152
A.2.4 Implementation Test Network 155
A.3 Test Cases 158
A.3.1 Test Case Descriptions 160
A.4 Implementation Testing Findings and Observations 162
A.5 Conclusion 163

Appendix B. RingBuilder™ Direct Cost Optimization via Specific Progress Example 164

Appendix C. RingBuilder™ netJ Baseline (Case23) and netM Baseline (Case26) Ring Designs 169

C.1 netJ (Metropolitan Network) Baseline: Case23 169
C.2 netM (Long-Haul Network) Baseline: Case26 170

List of Tables

Table 1: Summary of Operating Characteristics for Self-Healing Rings (adapted from [7]) 22
Table 2: Demand Bundling/Routing Options 30
Table 3: Experimental Networks 79
Table 4: Technology Cost Models 80
Table 5: Summary of Experimental Test Cases 82
Table 6: Test Suite 1 84
Table 7: Test Suite 1 Results 86
Table 8: Test Suite 2 87
Table 9: Test Suite2 Results 88
Table 10: Test Suite 3 89
Table 11: Test Suite 3 Results for netJ 99
Table 12: Test Suite 3 Results for netM 99
Table 13: Test Suite 4 100
Table 14: Test Suite 4 Results 101
Table 15: Test Suite 5 102
Table 16: Test Suite 5 Results 103
Table 17: Test Suite 6 104
Table 18: Test Suite 6 Results 107
Table 19: Test Suite 7 108
Table 20: Test Suite 7 Results 109
Table 21: Test Suite 8 110
Table 22: Technology Choices for Multi-Technology netJ Design 110
Table 23: Technology Choices for Multi-Technology netM Design 110
Table 24: Test Suite 8 Results 112
Table 25: Test Suite 9 113
Table 26: Design Option Selection 114
Table 27: Test Suite 9 Results 116
Table 28: Test Suite 10 117
Table 29: Test Suite 10 Results 118
Table 30: Test Suite 11 119
Table 31: Test Suite 11 Results 120
Table 32: Test Suite 12 121
Table 33: Test Suite 12 Results 122
Table 34: Interpretation of Test Suite Summary 123
Table 35: Conditional Compilation Diagnostic Options 151
Table 36: netZb Point-to-Point Demand Information 156
Table 37: netZb Span Information (with k-way Routing of Demands Span Cross-sections) 158
Table 38: Summary of Software Tests 159
Table 39: Software Tests by Network, System Technology and Design Considerations 160
Table 40: Design Example Cost Factors 166

List of Figures

Figure 1: Number of Possible Designs Generated vs. Number of Ring Candidates 5
Figure 2: Unidirectional Ring Demand Routing (adapted from [6]) 7
Figure 3: Bidirectional Ring Demand Routing (adapted from [6]) 8
Figure 4: SONET System Hierarchy [8] 9
Figure 5: Line vs. Path Switched Ring Protection Reactions (adapted from [6]) 10
Figure 6: Span reuse in a BLSR (adapted from [4]) 12
Figure 7: Bidirectional Line Switched Ring 13
Figure 8: 4 Fibre Bidirectional Line Switched Ring with Partial Span Cut 14
Figure 9: Unidirectional Path Switched Ring-- Span Failure Ring Reaction 15
Figure 10: Unidirectional Line Switched Ring- Span Failure Ring Reaction 16
Figure 11: Bidirectional Path Switched Ring - Span Failure Reaction 18
Figure 12: An example of BLSR ring loading (adapted from [4]) 20
Figure 13: An example of UPSR loading (adapted from [4]) 21
Figure 14: Network Topology Constraint Map 24
Figure 15: Working Path Placement Considerations 27
Figure 16: System Technology Considerations 30
Figure 17: Design Considerations 33
Figure 18: Ring Capacity Redundancy Measurement 36
Figure 19: Geographic Redundancy Measurement 36
Figure 20: Modular Geographic Redundancy Measurement 37
Figure 21: Balance Efficiency 39
Figure 22: Capture Efficiency 40
Figure 23: Ring Traffic Capture --Maximum Capture 41
Figure 24: Ring Traffic Capture -- Minimum Capture 41
Figure 25: Ring Traffic Capture -- Partial Capture 42
Figure 26: Nortel SONET Planner 47
Figure 27: Bellcore SONET Toolkit 49
Figure 28: RingBuilder™ 53
Figure 29: RingBuilder™ Iterative Ring Synthesis Framework 54
Figure 30: Cycle Finding Options 55
Figure 31: Demand Routing Options 56
Figure 32: Balance vs. Capture Optimization 60
Figure 33: Specific Progress 61
Figure 34: Single Technology Balance vs. Capture Optimization 63
Figure 35: Specific Progress Direct Cost Optimization Greedy Iterative Framework . 66
Figure 36: Example -- Network and OD Pairs 67
Figure 37: Example -- Shortest Path Demand Routing 67
Figure 38: Example -- Cycle Set 68
Figure 39: Example -- Ring Candidate Loading and Ring Selection for Iteration 1 68
Figure 40: Example -- Ring Candidate Loading and Ring Selection for Iteration 2 69
Figure 41: Example -- Ring Candidate Loading and Ring Selection for Iteration 3 70
Figure 42: Example -- Rings in Final Design 70
Figure 43: Design Considerations as Addressed by RingBuilder™ 71
Figure 44: netJ 73

Figure 45: netY 74
Figure 46: netYa 75
Figure 47: netYb 76
Figure 48: netYm 77
Figure 49: netM 78
Figure 50: netJ -- Cost vs. Alpha 90
Figure 51: netM -- Cost vs. Alpha 90
Figure 52: netJ -- Number of Transitions vs. Alpha 91
Figure 53: netJ -- Protection BW*Distance vs. Alpha 92
Figure 54: netM -- Protection BW*Distance vs. Alpha 92
Figure 55: netM -- Number of Transitions vs. Alpha 93
Figure 56: netJ -- Average Specific Progress vs. Alpha 94
Figure 57: netM -- Average Specific Progress vs. Alpha 94
Figure 58: netJ -- Number of Rings vs. Alpha 95
Figure 59: netM -- Number of Rings vs. Alpha 95
Figure 60: netJ -- ADM Count vs. Alpha 96
Figure 61: netM -- ADM Count vs. Alpha 96
Figure 62: netJ -- Total Cost, Span Cost and Node Cost vs. Alpha 97
Figure 63: netM -- Total Cost, Span Cost, and Node Cost vs. Alpha 97
Figure 64: netJ -- Average Balance and Capture vs. Alpha 98
Figure 65: netM -- Average Balance and Capture vs. Alpha 98
Figure 66: Sample Failed Demand 105
Figure 67: Demand Endpoints and Spans Containing Unrouted Demand Segments .. 106
Figure 68: Ring Loading Differences Between Case45a and Case45b 115
Figure 69: netJ -- Cost vs. Average Specific Progress 125
Figure 70: netM -- Cost vs. Average Specific Progress 126
Figure 71: netJ -- Node and Span Cost vs. Specific Progress 126
Figure 72: netM -- Node and Span Cost vs. Average Specific Progress 127
Figure 73: netJ -- Cost vs. Protection Bandwidth-Distance Product 128
Figure 74: netM -- Cost vs. Protection Bandwidth-Distance Product 128
Figure 75: netJ -- Cost vs. Installed Margin Bandwidth-Distance Product 129
Figure 76: netM -- Cost vs. Installed Margin Bandwidth-Distance Product 130
Figure 77: netJ -- Cost vs. Average Balance 131
Figure 78: netM -- Cost vs. Average Balance 131
Figure 79: netJ -- Cost vs. Average Capture 132
Figure 80: netM -- Cost vs. Average Capture 133
Figure 81: netJ -- Balance, Capture, and Specific Progress vs. Ring Number for Case23 Design 134
Figure 82: netM -- Balance, Capture, and Specific Progress vs. Ring Number for Case26 Design 134
Figure 83: netJ -- Cost, Progress, and Specific Progress vs. Ring Number for Case23 135
Figure 84: netM -- Cost, Progress, and Specific Progress vs. Ring Number for Case26 135
Figure 85: netJ -- Cost vs. Number of ADMs 136
Figure 86: netM -- Cost vs. Number of ADMs 137

Figure 87: netJ -- Cost vs. Number of Rings 138
Figure 88: netM -- Cost vs. Number of Rings 138
Figure 89: High Level RingBuilder™ Functional Flow 153
Figure 90: Sample of Manual Verification Spreadsheet 154
Figure 91: netZb: Test Network for Tests 155
Figure 92: Shortest Path Routes for all Demand Pairs in Validation Network 157
Figure 93: Cycle Set for Validation Network 157
Figure 94: Ring Cost Components 164
Figure 95: Ring Progress 165
Figure 96: Ring Costing 165
Figure 97: Design Example: Routed Demands and Cycle Set 166
Figure 98: Design Example: Iteration 1 167
Figure 99: Design Example: Iteration 2 167
Figure 100: Design Example: Iteration 3 168
Figure 101: Design Example: Iteration 4 168
Figure 102: Case23 Design 169
Figure 103: Case26 Design -- Rings 1 Through 16 170
Figure 104: Case26 Design -- Rings 17 Through 27 171

Nomenclature

Access-Limited System -- A ring system that can only add/drop or transit a portion of its optical line bandwidth at any one node.

Active Node -- A network node that sources or sinks demands.

Add/drop Restricted (node) -- A node that cannot add or drop demands.

ADM -- Add-Drop Multiplexer, the terminal equipment present at the active nodes in a ring system.

ADM Port Count Limit -- The maximum number of add/drop ports that can be used to source, terminate, or transit traffic at a node on a ring.

APS -- Automatic Protection System or 1+1 APS. A transmission system with a fully duplicated protection transmission path. It carries the same signals as the working path and has the capability to automatically switch to the protection path upon detection of failure of the working path.

Balance -- A measure of the working capacity carried by each ring span relative to the amount of protection capacity embedded in the ring.

BLSR -- Bidirectional line-switched ring.

Bundled Demands -- One or more demand management units of bandwidth that must be treated as a single entity.

Capture -- A measure of how well a ring contains the demands it is slated to carry.

Common Equipment -- Power supplies, equipment shelves and racks, electronic subsystems, and line optical interfaces of an ADM system.

Cycle -- An abstract term used in graph theory to denote a closed trajectory through a network. Each unique, closed, non-looping path constitutes a distinct cycle.

DCS -- Digital crossconnect system. An electronic version of the electrical patch panel used to interconnect electrical signals.

Demand -- An amount of bandwidth that has to be transported between nodes in a network.

Demand Bundling -- The aggregation of small local demand groups into atomic entities by strategically interconnecting them at local hubs.

Demand Management Unit -- The smallest unit of bandwidth which a system processes

as a unique entity.

Demand Segment -- The portion of a demand route that is present on a span.

Demand Route -- A set of sequential adjacent demand segments that are routed along physical spans from source to destination.

Demand Route Distribution -- A technique where demands are split equally over multiple equivalent shortest path routes. Demand route distribution is sometimes called *k-way routing* because the demands are split over 'k' equivalent routes. In contrast, routing a demand over a single shortest path route (even if there are other possible equally short routes) is called *1-way* routing.

Demand Splitting -- The ability to route different integer subunits of the demand bandwidth on the 2 opposite routes around the ring from the source to the destination nodes.

Distance-Weighted Balance -- The inverse of geographic redundancy.

DS1 -- Digital Signal Level 1. A multiplexed digital transmission format that has a capacity of twenty-four 64kb/s channels.

DS3 -- Digital Signal Level 3. A multiplexed digital transmission format that has a capacity of 672 DS1s.

Geographic Redundancy -- A modification of simple redundancy that weights the working capacity and the protection capacity on a ring by the geographical distance that each respectively travels. Also called *distance-weighted redundancy.*

Glass-Through Node -- An inactive node without regeneration equipment (Contrast: *pass-through node*).

Green-Field Design -- A rare opportunity where a network planner is able to specify the entire inter node connectivity structure of a network prior to designing a transport network architecture.

Hop Count -- A count of the number of spans on which the demand travels from source to destination.

Inactive Node -- A node that has no demand entry or exit activity.

Line -- The subset of a path that exists between 2 instances of line terminating equipment. A line is a concatenation of sections. (Contrast: *path* and *section*)

Line-Switched Ring - A ring with a recovery process that isolates only the line that is

affected in the path when a ring path is broken. (Contrast: *path-switched ring*).

Link -- A demand management unit of transport bandwidth within a span of a ring system.

LTE -- Linear Terminal Equipment (also referred to as Lightwave Terminating Equipment), an element used as terminal equipment at active nodes in mesh and sometimes in ring-based networks.

Margin -- The unused working capacity of a transmission system.

(Fully) Mesh-Connected Network -- A network with each node connected to every other node.

Minimally-Connected Network -- A 2-connected network.

Modular Geographic Redundancy -- A modification of geographic redundancy that considers any unused working capacity in the modular system.

Modularity -- The discrete capacity of a system in terms of the basic demand management unit.

Modularized Distance-Weighted Balance -- The inverse of modularized geographic redundancy.

Nearest-Neighbor Connected Network -- A network with all nodes connected to their nearest physical neighbors.

Nodal Degree -- The number of spans connected to a node.

Node Cost -- The cost of the ADM equipment placed at active nodes and the repeaters placed at pass-through nodes of a ring. ADM cost is comprised of the cost of the ADM common equipment and the ADM provisionable equipment.

Node-Oriented Constraints -- The equipment add/drop restrictions and transition restrictions at a node.

Non-Access-Limited System -- A transport system that can add, drop, or transit all of its bandwidth at a single node.

OC12 -- Optical Carrier Level 12. A SONET transmission format that has a capacity of 12 STS1s.

OC48 -- Optical Carrier Level 48. A SONET transmission format that has a capacity of 48 STS1s.

OC192 -- Optical Carrier Level 192. A SONET transmission format that has a capacity of 192 STS1s.

Origin-Destination (OD) Pair -- A set of nodes which mutually source and sink demands.

Pass-Through Node -- An inactive node that has a regenerator (contrast: *glass-through node*)

Path -- The entire collection of equipment and fibre between the end nodes of a demand. A path is a concatenation of *lines*. (Contrast: *line* and *section*)

Path-Switched Ring -- A ring with a recovery process that isolates the entire affected path from entry node to exit node.

Progress -- A measurement of the amount of demand carried on a ring.

Recovery Process -- A ring's survivability reaction to a ring failure.

Provisionable Equipment -- The tributary access cards which are provisioned on an ADM or DCS as required for adding or dropping demands or for transiting demands to other rings.

Redundancy -- A measure of the amount of protection capacity that has to be deployed relative to the used working capacity present in a system or network design.

Regenerator -- An element used in optical transmission systems to amplify the optical signal.

Ring -- A group of nodes on a cycle interconnected by a series of transmission systems following the path of the cycle. The ring may be *unidirectional* or *bidirectional*, depending on whether the signals carried from ring entry node to ring exit node are carried one way or both ways around the ring.

Ring System Technology Type -- A description of the ring system's optical line rate, its native demand management unit, its protection switch type (i.e. line switched or path switched), and its fibre count (2 or 4 fibre).

Ring Topology -- A ring's path through the nodes and spans of the network.

Route -- A trajectory through the spans and nodes of a network.

Section -- A fibre segment between inactive nodes or between an active node and an inactive node. Many sections can be concatenated to form a *line* and many lines concatenated to form a *path*. In turn, the superposition of many paths tracing out portions of a common cycle forms a *ring*.

Self-Healing -- The ability of a transmission system to automatically recover from a fault.

In general, the term *self-healing* is associated with a class of mesh networks which can autonomously reconfigure themselves under failure conditions. In this thesis, self-healing is used in the context of individual ring systems autonomously reconfiguring when a failure occurs.

Shortest Logical Path Routing -- A routing technique that minimizes the hop count of the demand routes.

Shortest Path Routing -- A routing technique that minimizes either the total hop count or the total physical distance a demand has to travel from source to destination.

Shortest Physical Path Routing -- A routing technique that minimizes the physical distance of the demand routes.

Simple Redundancy -- The sum of the installed protection capacity divided by the protected used working capacity across all spans on a ring.

Span -- The set of all links between 2 nodes of a network. Multiple ring systems may be on a span.

Span Addition - The modification of existing network connectivity by specifying additional inter-node connection(s).

Span Elimination -- The modification of existing network connectivity by disusing existing inter-node connection(s).

Span Reuse -- The ability to assign a time slot to different signals on non-coincident paths around a bidirectional line-switched ring.

Sparsely-Connected Network -- A network that has some 2-connected nodes, but also some nodes of higher degree.

Specific Progress -- A measure of the amount of progress towards design completion the ring system under evaluation contributes, relative to the cost of implementing that system.

STS1 -- Synchronous Transport Signal Level 1. A 51.84 Mb/s signal.

STS3c -- A 155.52 Mb/s data stream.

Survivability -- The ability of a transmission system to rapidly recover from a partial or complete transport span failure.

Transition Constrained Node -- A node that does not support the passing of signals from one ring system to another.

Tributary -- An add/drop signal from an ADM. The tributary bandwidth is an integer multiple of the demand management unit of the ADM.

UPSR -- Unidirectional Path-switched ring.

2-Connected Network -- A network where every node has at least 2 spans connected to it.

1. Introduction

1.1 Design of Transport Networks

The migration of the world towards an information society is resulting in an increasing dependence on the telecommunications infrastructure. The availability of the network as well as its information carrying capacity are key factors governing its usefulness. In addition to the growing utilization of traditional telephony applications, the proliferation of broadband multimedia services continues to stress both transmission and switching payload carrying capabilities. Telecommunications transport networks provide the high bandwidth conduits which interconnect switching centres to facilitate communication on a city-wide, continental, and global scale. These networks provide the backbone on which circuit-switched and packet-switched infrastructures are built. The medium of choice today for the bulk transport of high bandwidth payloads is digital fibre optics. Current system rates are 10 Gb/s (130,000 voice channels) per fibre pair and there can be more than 50 fibre pairs per cable. Dense Wave Division Multiplexing (DWDM) can multiplex 32 10 Gb/s optical channels, resulting in 320 Gb/s on a single fibre. The use of such high-capacity transport systems greatly increases the vulnerability of the network to single system failures.

The ability of a system to rapidly recover from a partial or complete transport span failure is extremely important in maintaining network integrity. This ability is defined as *survivability*. If the system can automatically recover from a fault, it is said to be *self-healing*. Cost-effective deployment of bandwidth is critical for the communications supplier due to decreasing operating margins in today's competitive environment. The aim of the network architect is to design cost-effective *survivable* networks that have high bandwidth utilization *efficiency*.

Three main survivable network topologies have been identified to date: point-to-point with diversely routed protection (1+1 DP), mesh, and ring [1]. A 1+1 DP system is a point-to-point transmission system which connects an origin-destination pair of nodes using two diverse routes through a network. A *route* is a trajectory through the spans and nodes of a network. If one route fails, another can take its place, ensuring survivability of the demand transport. An *origin-destination(OD)* pair is a set of nodes which mutually

source and sink demands. A *demand* is data that has to be transferred between two nodes. It is assumed that the amount of demand in an OD pair is the same in both directions.

In a mesh network, the nodes are interconnected with point-to-point transmission systems. Mesh networks have the property that multiple routes can be formed between all OD pairs. If a communications path fails, a mesh network can usually be reconfigured to re-route the affected demands, ensuring survivability. Mesh networks have the advantage of extreme capacity efficiency, but a major challenge that currently exists is the cost-effective implementation of full-mesh infrastructures due to the cost of the node equipment. In general, the term *self-healing* is associated with a class of mesh networks which can autonomously reconfigure themselves under failure conditions. In this thesis, self-healing is used in the context of individual ring systems autonomously reconfiguring when a failure occurs.

A ring network is comprised of self-healing rings (SHRs). A *ring* is formed when a group of nodes on a cycle are interconnected by a series of transmission systems following the path of the cycle. The ring may be *unidirectional* or *bidirectional*, depending on whether the signals carried from ring entry node to ring exit node are carried one way around the ring or both ways around the ring. A ring-based network is constructed by choosing rings from the set of available cycles for the network. A *cycle* is not a physical entity. It is a term used in graph theory to denote a closed trajectory through a network. Each unique, closed, non-looping path constitutes a distinct cycle. *Unique* in this context means that the span sets describing each cycle differ by at least one span. *Closed* means that the path which is traced starts and ends on the same node. *Non-looping* means that every node is visited once and only once. Upon detection of a communications failure, a self-healing ring automatically switches the demands it carries to a diverse communications route. The protection route lies along the opposite trajectory of the cycle upon which it is based, isolating the failed route. Ring systems can recover from signal path failures in 50 ms, at least 2 orders of magnitude faster than the response time of current mesh restorable networks. However, this capability of ring systems comes at the expense of having substantially more dedicated protection capacity than mesh restorable networks. However, node equipment for ring networks is currently substantially less expensive than for mesh networks, and thus ring networks are more cost-effective to implement.

Ring networks are currently very popular in industry, since they provide an economic trade-off between fault recovery time and system bandwidth requirements.

1.2 Design Problem Complexity

The requirement to build efficient survivable ring-based networks has generated considerable interest in industry for the development of tools and methodologies to assist in the design of transport networks based on multiple rings. The operation and provisioning of isolated ring systems is straightforward, and they are only incrementally more difficult to design than a conventional non-survivable transport system. The problem becomes substantially more complex when one considers the design and deployment of a network of multiple self-healing rings. To date, this problem has largely been left to the network designer. Some computer-based tools have been developed to analyze designs generated by the planner. However, the quality of designs generated with these analysis tools depends largely on the skills, experience, and heuristics that the planners have at their disposal. As a result, these designs can vary widely in their quality and efficiency.

Another approach to the problem, automated synthesis, has been the subject of considerable research. Due to the complexity of multiple ring design, this method has not yet been generally perfected. This complexity is a function of the large number of degrees of freedom present. Degrees of freedom exist at the network level, the system level, and at the design level.

The planning, operation, and deployment of single ring systems is relatively simple. The design problem degenerates to ensuring that the SONET ring limit of a maximum of 16 active nodes is obeyed [2]. The ring bandwidth is then appropriately selected, and the demands are routed on the ring. A local area network SONET ring would be deployed in this manner. Unfortunately, it is usually inefficient or impossible to design a network so that a single ring can carry all of the traffic between all of the nodes. In general, multiple rings have to be deployed to transport the bandwidth.

The number of potential cycles for a network is bounded by [3]:

$$cycles \leq 2^{s-n+1} \tag{1}$$

where s is the number of spans in the network and n is the number of nodes in the network.

A design is developed by choosing cycles, one at a time, from this set until all of the demands have been contained in the chosen cycles. The order in which the cycles are chosen is important. As each cycle is selected, a subset of the network demands is associated with that cycle. If the cycles are chosen in a different order, it is likely that each cycle will carry different demand segments, and a different design will result. A *demand segment* is the portion of a demand route that is present on a span. A *demand route* is a set of sequential adjacent demand segments that are routed along physical spans from source to destination. The number of ways in which a ring-based network may be created is thus a combinatorially complex problem. Once a cycle has been chosen, various nodes must be selected to determine the payload that the cycle can carry. Some nodes will be designated to be members of OD pairs, and be designated active nodes, whereas others will not source or terminate any demands and will just be traversed by demands going from origins to destinations. This sub-problem is combinatorial in nature. As a result of these permutations and combinations, the size of the design space becomes:

$$D = \left(\sum_{i=1}^{c} \frac{c!}{(c-i)!} \right) \times \left(\sum_{j=2}^{y_i} \frac{y_i!}{j!(y_i-j)!} \right) \tag{2}$$

where D is the number of designs, c is the number of ring candidates (cycles) for a network, y_i is the number of nodes in candidate cycle i, and j is the number of active nodes within cycle i. Because this function grows very rapidly, it is virtually impossible to exhaustively examine the entire solution space, even for a network of moderate size. The ability to efficiently find a cost-effective solution to this problem is key to an efficient ring network design technique.

A conservative lower bound for the above equation can be determined by assuming that only one realizable ring candidate exists for each cycle discovered. In this case, the solution space grows for each new cycle discovered as shown in Figure 1.

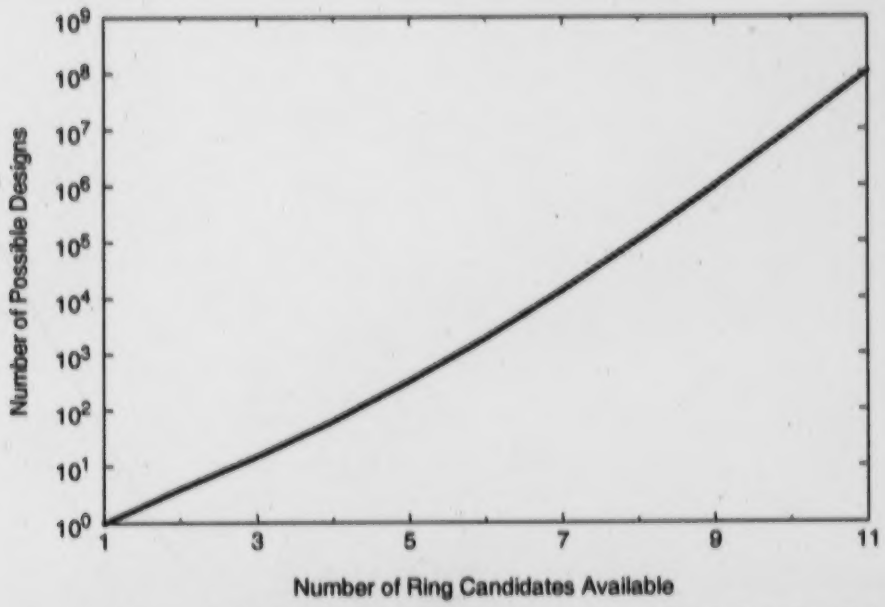


Figure 1: Number of Possible Designs Generated vs. Number of Ring Candidates

The curve shown in Figure 1 is described by:

$$D = \sum_{i=1}^{c} \frac{c!}{(c-i)!} \tag{3}$$

where D is the number of possible designs, c is the number of cycles available, and i is the number of cycles in the design.

In general, far fewer *realizable* designs exist than dictated by equation 2 or even equation 3, since not all permutations of ring candidates can accommodate all of the network demands. The problem is far more complex than equation 3 suggests, since a cycle alone does not completely specify a ring. The actual components of the ring add another level of combinatorial complexity to the problem specification.

The loading of demands onto a ring candidate is a subproblem which must be addressed when a set of ring candidates is created in preparation for ring selection. The problem of loading demands onto a bidirectional line switched ring (a ring type which will be described in the next chapter) without splitting has been proven to be an NP-complete problem [4]. *Demand splitting* in this context is defined the ability to route different integer subunits of the demand bandwidth on the 2 opposite routes around the ring from the

source to the destination nodes. The ring design problem complexity is at least as great as that of the subproblems comprising it. Therefore, ring network design is a very complex problem indeed.

1.3 Research Objectives

The intent of the current work is to study the many aspects of the multiple ring design problem and to develop an automated approach to the design of such networks. This thesis tests two main ring selection hypotheses in the multiple self-healing ring network design problem using a newly developed heuristic framework called RingBuilder™. The first hypothesis is ring selection via a hybrid balance vs. capture evaluation rule. The second is ring selection via specific progress. RingBuilder™ is also used to explore various dimensions of the ring network optimization space, including different working path placement options, system technology options, and design considerations. Because the complex demand routing subproblem is such an important component of the synthesis process, several demand sorting and loading heuristics are introduced and tested.

1.4 Thesis Organization

Chapter 1 has introduced the concept of a telecommunications transport network, the main architectural approaches used to make the transport network survivable, and the advantages that ring-based network design has over the other architectures. The complexity of the ring network design problem was discussed, and the main research objectives of this work were introduced. **Chapter 2** provides a tutorial introduction to the two main types of self-healing rings used in telecommunications networks. **Chapter 3** describes the design principles and issues surrounding the design of multiple self-healing ring networks and will discuss the main existing approaches to that design problem. **Chapter 4** introduces RingBuilder™, a new multiple self-healing ring network design synthesis system, and describes it at a conceptual level. **Chapter 5** describes the experiments conducted with RingBuilder™ to test the two main ring selection hypotheses under a variety of network, system, and technology constraints. **Chapter 6** summarizes the results obtained, and provides potential future research directions based on this work.

2. Types of Self-Healing Rings

2.1 Introduction

This chapter provides a tutorial introduction to the self-healing ring (SHR), the primary building block used to construct multiple SHR networks. First, necessary terms are introduced so that the reader can better understand the subsequent discussion of the main types of SHRs. Then, for completeness, the eight possible types of SHRs are introduced. Three of these ring types, which are formally standardized in the SONET Generic Recommendations, are then described in more detail [2,5]. These are the only types of practical concern, and the experimental trials described in Chapter 5 are based on the use of these standardized ring types.

2.2 Relevant Ring Concepts and Terminology

To make a ring survivable, a duplicated, geographically diverse path must exist for every signal carried from every entry node to every exit node for all possible single path failures. With a unidirectional ring, the protection bandwidth allocated for the working signals is transported in the opposite direction to the working bandwidth. In Figure 2, the transport path from node 1 to node 3 is via node 2 (path = 1,2,3), while the return path from node 3 to node 1 is via node 4 (path = 3,4,1). The protection path for (1,2,3) is (1,4,3), while for (3,4,1) it is (3,2,1).

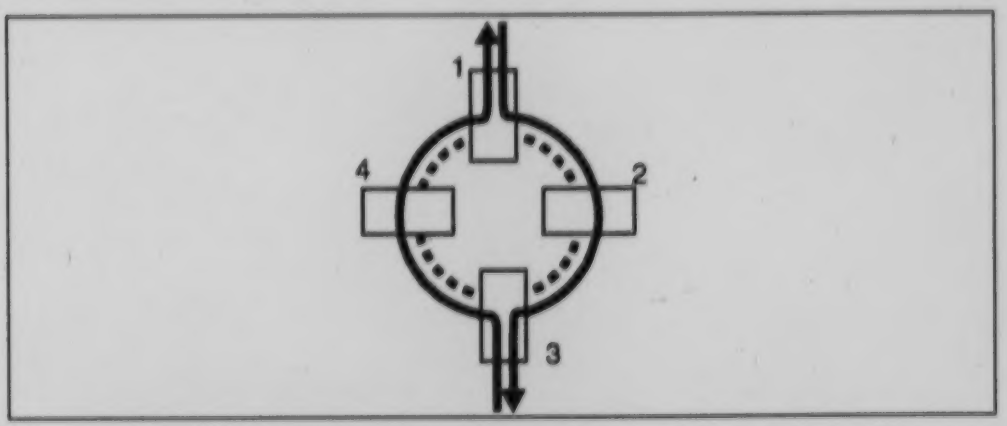


Figure 2: Unidirectional Ring Demand Routing (adapted from [6])

In contrast, in a bidirectional ring both the forward and return paths are routed on the same set of nodes (for transport from node 1 to node 3 the path is (1,2,3); for transport from node 3 to node 1 the path is (3,2,1)). Bidirectional rings have the advantage of an equal delay in both the forward and return paths; this is not the case with unidirectional rings. An example of the demand routing on a bidirectional ring is shown in Figure 3.

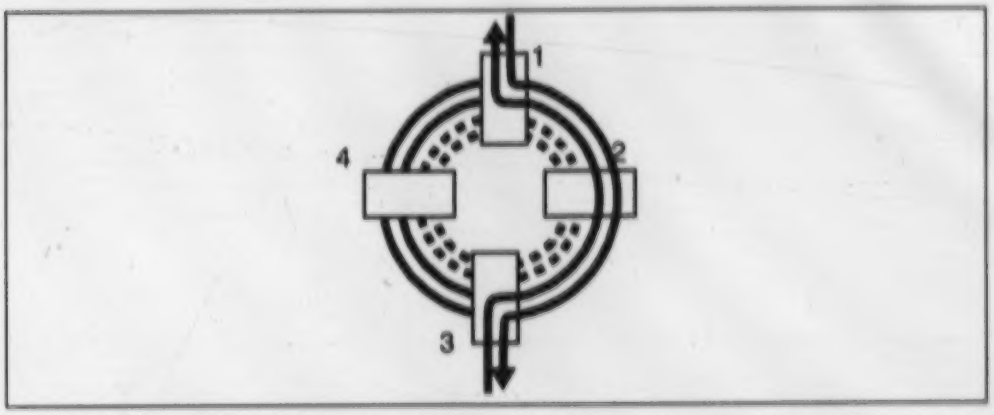


Figure 3: Bidirectional Ring Demand Routing (adapted from [6])

The survivability reaction to a ring failure is called the *recovery process*. The ring recovery process can take one of two forms: *line switching* or *path switching*. In order to understand these terms, it is necessary to define what is meant by *line* and *path* in the context of a generic SONET transmission system.

The SONET standard was originally developed for point-to-point systems. The ring topology equipment interconnection was enabled by the hierarchical protection structure first implemented in the SONET standard, as well as by diversely routed protection bandwidth, first implemented in point-to-point transport systems. The SONET standard specifies protection communications bandwidth and actions at the section, line, and path level.

Consider the simple transmission system outlined in Figure 4. The entry and exit nodes are the end nodes of a path in a SONET transmission system. Thus the *path* consists of the entire collection of equipment and fibre between the end nodes. The *line* is the subset of the path that exists between 2 instances of line terminating equipment. Line terminating equipment exists at all active nodes. An *active node* is one where demand entry or exit activity occurs. In contrast, a node that does not have any demand entry or exit activity

is an *inactive node*. An inactive node is called a *glass-through* if it has no regeneration equipment, or a *pass-through* if it does have a regenerator[4]. A *regenerator* is used in optical transmission systems to amplify the optical signal. Each segment between inactive nodes or between an active node and an inactive node is called a *section*. Many sections can be concatenated to form a line and many lines concatenated to form a path. In turn, the superposition of many paths tracing out portions of a common cycle forms a ring.

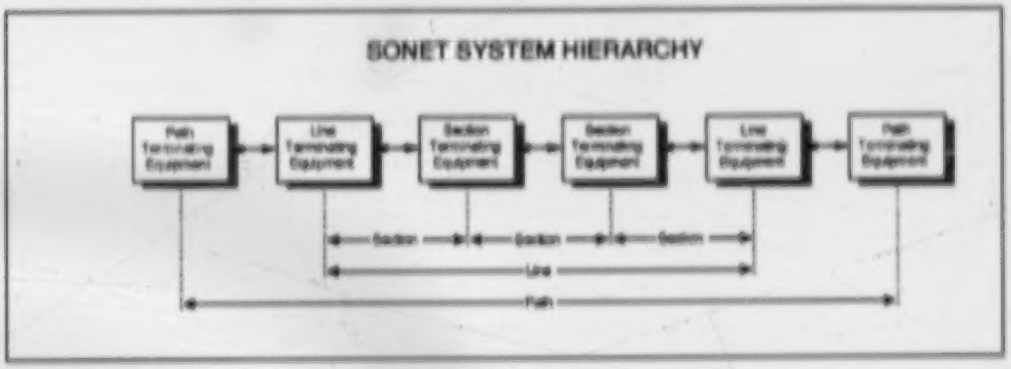


Figure 4: SONET System Hierarchy [8]

A ring is said to be *line-switched* if the ring recovery process isolates only the line that is affected in the path when a ring path is broken. If, instead, the ring recovery process isolates the entire affected path from entry node to exit node, it is said to be *path-switched*.

Line switched rings switch from working to protection bandwidth at both active nodes adjacent to the failed span. These two active nodes must communicate with one another in order to coordinate the failure recovery. Line switched rings require each node to have knowledge of the entry nodes, exit nodes, and routing of all demands carried on the ring.

The protection mechanism for path switched rings is simple. It is essentially the same mechanism that exists for point-to-point protected transport systems (i.e. 1+1 APS). A *1+1 APS* (Automatic Protection System) is a transmission system with a fully duplicated protection transmission path that carries the same signals as the working path, and has the capability to automatically switch to the protection path upon detection of failure of the working path. Path switched rings only require exit node switching; the demands are permanently transmitted on both the working and protection routes at the entry node end.

Protection switching activity in path switched rings requires no global knowledge of source, destination, or route of the demands carried by the ring. The protection switching actions only require local analysis of the relative quality of the working and protection signals at the exit node end. A comparison of line vs. path switched activity is shown in Figure 5.

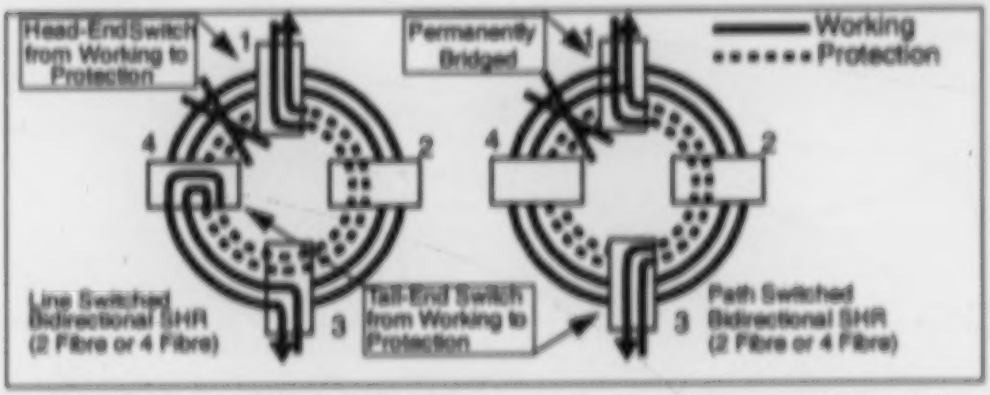


Figure 5: Line vs. Path Switched Ring Protection Reactions (adapted from [6])

A ring may have a 2 or 4 fibre cross section along its path. If a ring has 2 fibres it is called a *2-fibre ring*, with one fibre dedicated to clockwise transmission of signals, and one dedicated to counter-clockwise transmission. A ring with 4-fibres is called a *4-fibre ring*, with 2 fibres dedicated to clockwise signal transmission and 2 to counterclockwise transmission. This distinction is of special functional significance in the 4 fibre bidirectional line switched ring, described below.

2.3 Types of SHRs

Three main degrees of freedom exist in specifying an SHR: whether the ring is unidirectional or bidirectional, whether it is line-switched or path-switched, and whether it has 2 fibres or 4 fibres. As a result, there are eight possible types of self-healing rings. All 8 types will be described and it will be shown that only three of these subtypes are of practical concern (the 2- and 4- fibre bidirectional line switched ring and the 2 fibre unidirectional path switched ring).

2.3.1 Bidirectional Line Switched Rings (BLSRs)

Bidirectional Line Switched Rings (BLSRs) transport demands in both directions around the ring, and protect against node and span failures by line-level protection switching rather than by completely duplicated routing of demands from source to destination. The BLSR has equal transmit and receive path lengths for demands. Because the protection switching mechanism is line switching, the nodes surrounding the failure must act in concert to switch the bandwidth onto the protection channels. In order to prevent misconnecting traffic, all nodes in the ring must have a map of all add-drop locations as well as transport paths.

Because of the bidirectional transmission, the use of line switching, and the transmission of signals as time-division multiplexed entities, BLSRs do not make use of working and protection bandwidth all the way around the ring for each demand. Thus *span reuse*, the ability to assign a time slot to different signals on non-coincident paths around the ring, is possible. This capability makes BLSRs more efficient than other classes of rings in terms of overall carrying capacity at a given line rate. An example of span reuse is shown in Figure 6. In this figure, a demand uses time slot 1 of the BLSR ring along the counterclockwise path from node 1 to node 4. A second demand uses the same time slot in the clockwise path from node 1 to node 3 and a third demand uses this time slot in the clockwise path from node 3 to node 4. The same time slot is reused for three different demands on the ring.

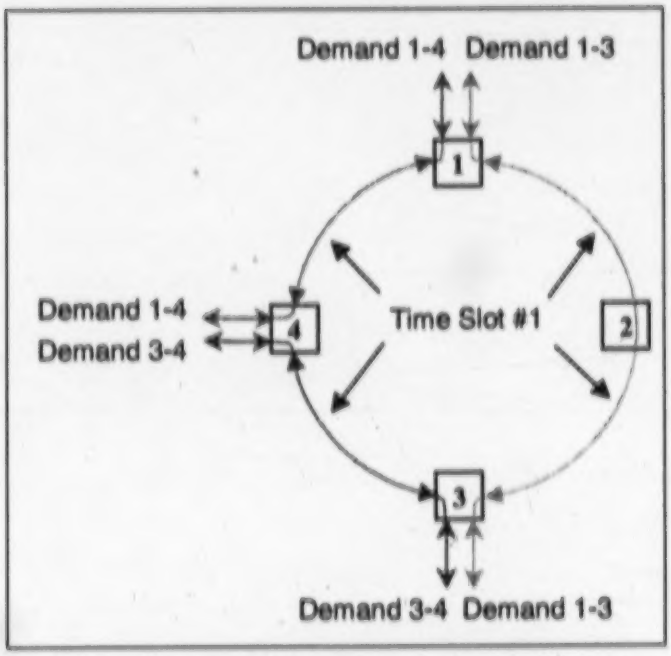


Figure 6: Span reuse in a BLSR (adapted from [4])

BLSRs are also known as 'shared protection' rings because the protection bandwidth is not automatically assigned for the working bandwidth carried. The protection bandwidth is utilized only when a fault causes a protection switch to occur. Because of this, the protection bandwidth can be used to carry extra traffic when it is idle. *Extra traffic* is bandwidth which is sometimes sold by telecommunications carriers to their customers at large discounts because it can be interrupted any time a self-healing ring reaction is required to protect against a failure.

Figure 7 shows the reaction of a 2-fibre BLSR to a span failure. Switching occurs at the nodes immediately adjacent to the failed span. The demand is back-hauled as it is switched onto the protection bandwidth at nodes B and C, and as it is switched from the protection bandwidth back onto the working bandwidth at nodes C and B. Control signalling is required between nodes B and C to coordinate the head- and tail- end switching processes.

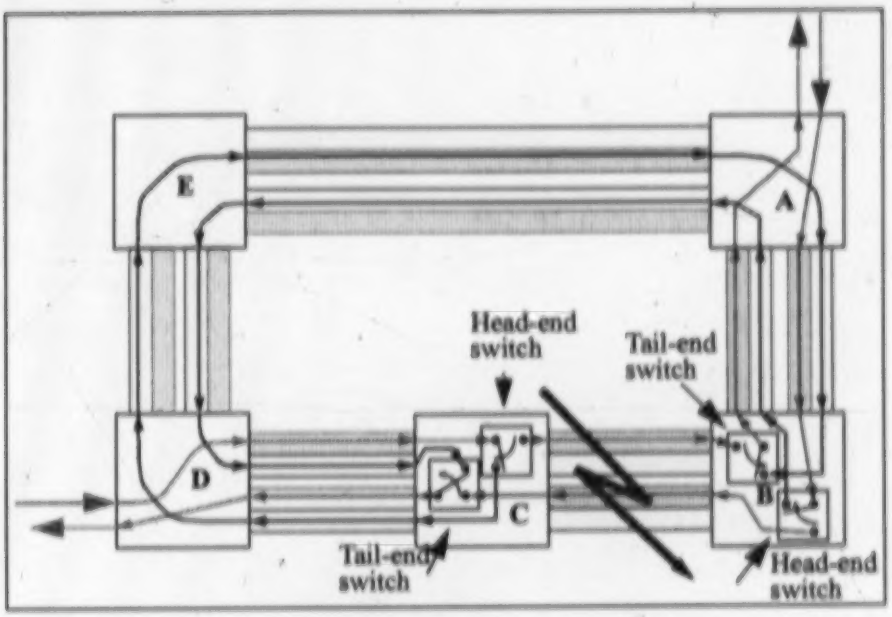


Figure 7: Bidirectional Line Switched Ring

The 4-fibre version of a BLSR has an additional fault protection mode which is not available in any of the other ring topologies: 1+1 span protection. If a partial span cut occurs, or more typically, a single fiber electronics failure occurs on an optical interface, a span protection mechanism can switch all of the working bandwidth onto the protection bandwidth. *Span protection* is a method by which the protection bandwidth of the span where the working signal failure occurs is used as the reroute path, rather than the physically diverse route on the other side of the ring. If, subsequently another failure occurs, either the span protection is dropped (freeing up protection bandwidth for the standard ring protection) or the ring protection mechanism cannot be used. This span protection mechanism is not available in the 2-fibre BLSR because of the logical instead of physical division of working and protection bandwidth in the 2 fibre ring. The 4-fibre BLSR has the additional advantage of physical separation of the working and protection bandwidth on separate fibres. This simplifies operation and maintenance of the system. For instance, in a non-failure condition, a technician can test the protection optical link without impact on any working traffic. The reaction of a 4-fibre ring to a partial span failure is shown in Fig-

ure 8.

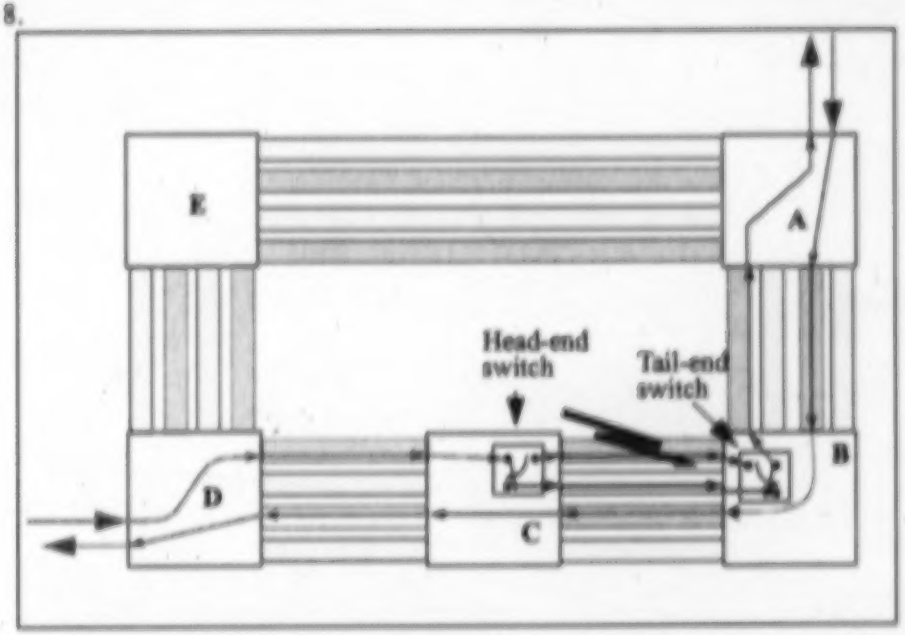


Figure 8: 4 Fibre Bidirectional Line Switched Ring with Partial Span Cut

2.3.2 Unidirectional Path Switched Rings (UPSRs)

Unidirectional path switched rings transport demands in only one direction around the ring, with protection bandwidth for the demands allocated in the opposite direction. Consequently, a bidirectional connection between 2 nodes on a UPSR utilizes bandwidth all the way around the ring. The transmit path from source to destination is on one set of spans and the return path is on the disjoint set of spans. Together, the transmit and receive paths traverse every span of the cycle. The protection bandwidth for a UPSR is contained in a transmission path that flows in the opposite direction relative to the working bandwidth. Working and protection bandwidth is allocated together on a UPSR since the entry node of a demand transmits on both the working and protection bandwidth simultaneously. Inter-node communication is not required for protection switching in a UPSR. In the event of a span or node failure, protection switching occurs at the exit node of the demand only. The decision to initiate a protection switch is based purely on relative receive signal quality of the working and protection paths. The protection switching mechanism replaces the entire path rather than just the failed span.

UPSRs do not allow bandwidth reuse since every unit of working capacity utilizes

bandwidth on both the working and protection channels all the way around the ring.

Figure 9 shows the reaction of a UPSR to a span failure. Protection switching occurs autonomously at the terminal nodes of the demands, without regard to events at other nodes on the ring. Because the bridging occurs at the entry of the demand and the protection switching occurs at the exit node, back-hauling of the demand bandwidth is not required.

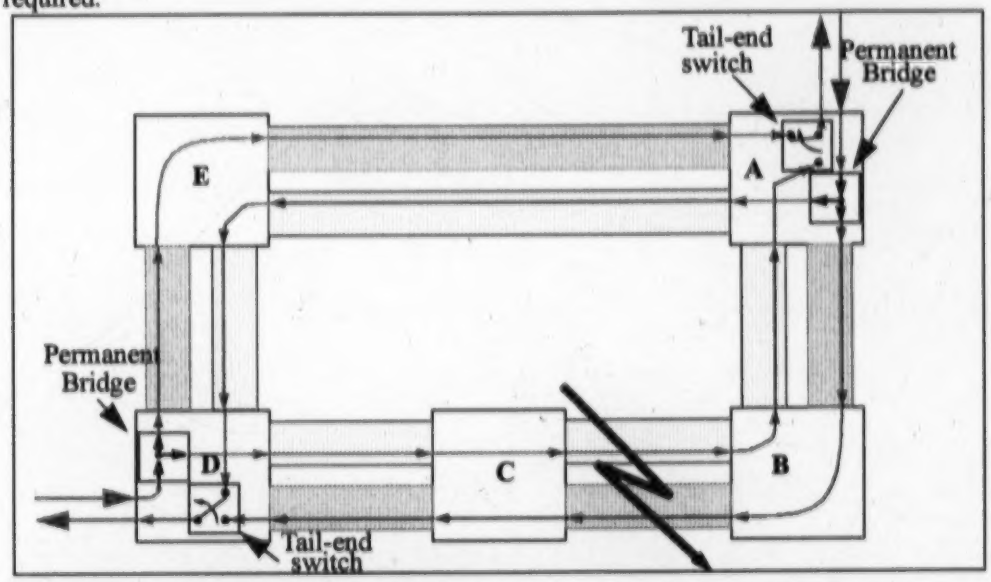


Figure 9: Unidirectional Path Switched Ring-- Span Failure Ring Reaction

2.3.3 Unidirectional Line Switched Ring

Like UPSRs, unidirectional line switched rings (ULSRs) transport demands in only one direction around the ring, with protection bandwidth for these demands allocated in the opposite direction. The protection switching mechanism, however, is like that of BLSRs in that only the failed span rather than the entire path is replaced by protection bandwidth. The switching activity occurs at the nodes adjacent to the failure rather than at the termination points of the affected demands. This switching activity must be coordinated and there must be communication between the two nodes. Further, each of the nodes must have a network map image to properly switch traffic between protection bandwidth and working bandwidth during the protection switching process. The network map is used to prevent traffic misconnection.

ULSRs do not provide any capacity benefits over the equivalent UPSRs, and their protection switching is more complicated. They do have the capability of carrying unprotected extra traffic on the protection bandwidth. However this extra traffic may have to be dropped in order to react to a ring or span failure affecting the working bandwidth. ULSRs have unequal transmit and receive delays for bidirectional demands due to the diverse transmit and receive trajectories around the ring.

In Figure 10, a 2-fibre ULSR with a failed span is shown. Nodes 'C' and 'D' immediately adjoining the failed span react by switching the affected signal to/from the protection path on the protection fibre. The line switching function causes the demand to be 'back-hauled' to the original source node 'A', as it makes its way along the protection path to the other end node 'D' of the failure. In doing so, the demand passes through the destination node 'E' on the protection path, only to be switched back onto the working path at node 'D' for delivery to the terminal node 'E'.

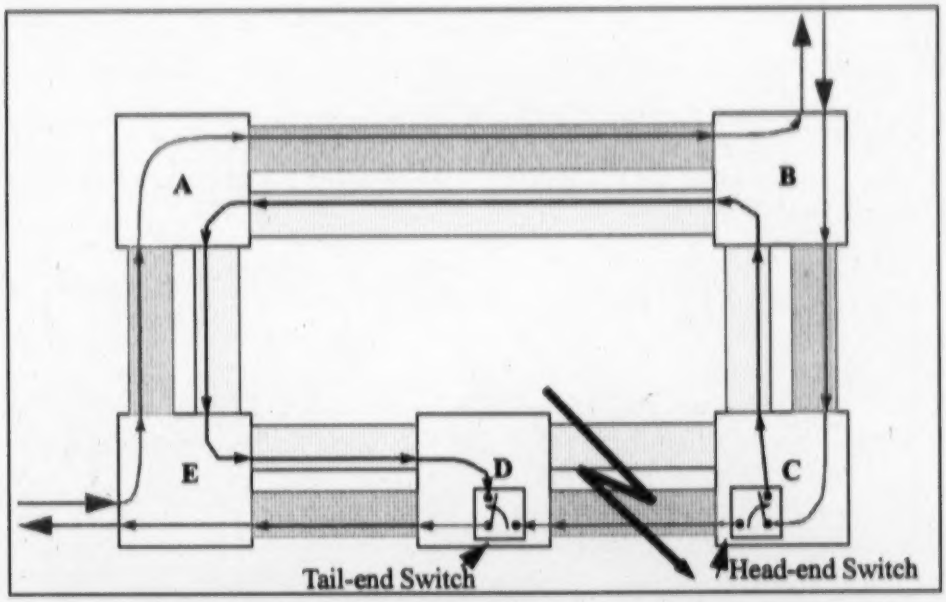


Figure 10: Unidirectional Line Switched Ring- Span Failure Ring Reaction

2.3.4 Bidirectional path switched ring

Bidirectional path switched rings (BPSRs) transport demands in both directions around the ring. The corresponding protection bandwidth travels in the opposite direction

to the working bandwidth. BPSRs, like UPSRs, permanently bridge the working traffic at the source node point, and depend on the sink node to continuously evaluate the quality of the working and protection streams, switching from the working to protection bandwidth in the case of a signal failure. Because the decision point for protection switching is isolated to the sink node, inter-node communication is not required, simplifying both the hardware and the software required to implement the system.

BPSRs do not have the capability to 're-use' bandwidth because of the permanent assignment of working and protection bandwidth for each demand all of the way around the ring. They don't provide any bandwidth carrying advantages over UPSRs operating at the same rate. However, BPSRs do provide equal transport delays in both the transmit and receive directions. In addition the 4 fibre BPSR provides physical separation of the working and protection bandwidth.

Figure 11 shows a 2-fibre BPSR reacting to a span failure. As with the UPSR, permanent head-end bridging is carried out for each of the demands. When a span failure occurs, the terminating end node of the demand switches from the working path to the protection path. Back-hauling is not required because of the head-end bridging at nodes A and D and tail-end switching at nodes D and A. Each fibre transports data in opposite directions around the ring. Note that in the 2BPSR, each of the fibres is logically partitioned into working and protection bandwidth, as each fibre supplies both working and protection bandwidth.

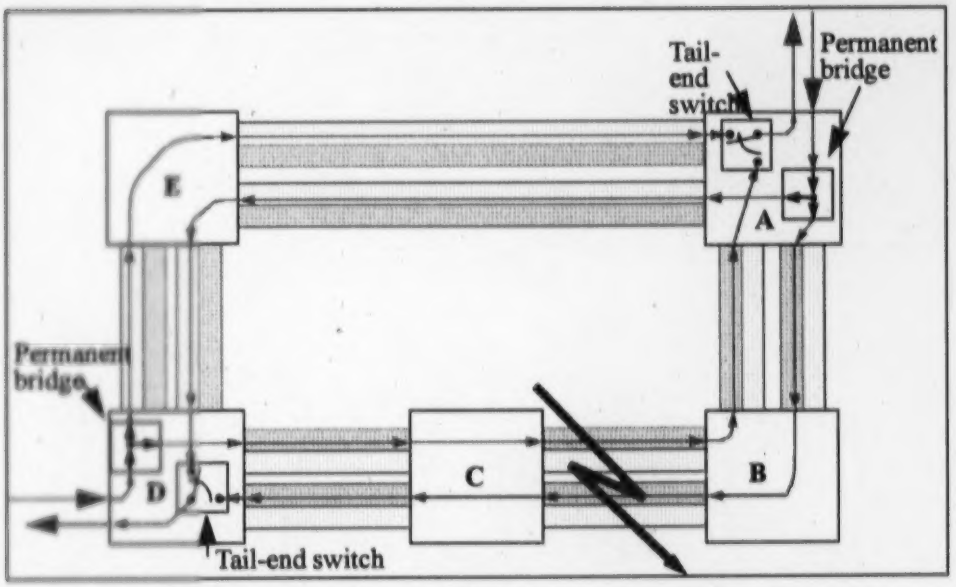


Figure 11: Bidirectional Path Switched Ring - Span Failure Reaction

2.4 Ring Loading

This section describes the ring loading task. The three SONET standard ring architectures are considered: 2-fibre BLSR (2BLSR), 4-fibre BLSR (4BLSR) and 2-fibre UPSR (2UPSR) as these are the ones that are actually produced by telecommunications equipment providers. The ring loading concepts introduced here are integral to understanding the multi-ring design problem in the next chapter.

2.4.1 BLSR System Capacity

BLSR system capacity is described in terms of the optical line rate of the transmission system as well as the demand management quantity. The *optical line rate* is the bit rate of the optical stream including the framing, overhead, and demands carried by the system. For SONET, standard optical line rates are expressed in terms of Optical Carrier n (OCn) values. The standard includes OC3 (155.52 Mb/s), OC12 (622.08 Mb/s), OC48 (2.49 Gb/s), OC192 (9.95 Gb/s), and OC768 (39.81 Gb/s). The *demand management quantity* is the unit of capacity which a transmission system can carry in each of its time slots between two nodes. For a SONET ring, the demand management quantity may be a

DS1 (24 64kb/second channels, 1.544 Mb/s), a DS3 (28 DS1s, or 672 64 kb/s channels, 44.736 Mb/s), an STS-1 (51.84 MB/s), an STS-3c (155.52 Mb/s), among others. Thus, an OC-48 system can have a system capacity of 1344 DS1s, 48 DS3s, 48 STS-1s, or 16 STS3c, depending on the nature of the Add Drop Multiplexer (ADM) network node equipment. Significant characteristics of ADMs are described in a subsequent section.

BLSR system capacity is defined and assigned on each line joining the active nodes. If a demand has to traverse several lines from the entry node to the exit node on a ring, sufficient remaining capacity for the demand must exist on each of the lines in the path. Otherwise the demand may only be partially carried, or may not be carried at all on that BLSR. A ring loading example using an OC-12 4-fibre BLSR (4BLSR12) is shown in Figure 12. Here, a set of STS-1 demands, presented in an demand matrix, is to be loaded onto the BLSR. The *demand matrix* is an upper triangular matrix which describes the amount of point-to-point demand that exists between end nodes on the ring. In order for the demand to be loaded on to the ring, it must be routed on the ring. By specifying the route that the demand will take around the ring from entry node to exit node, time slots on each line between active nodes can be assigned. The routing decisions are critical to the efficient utilization of the available bandwidth on the ring. The example in Figure 12 shows two possible routing arrangements for the given demand pattern. Arrangement (a) is not realizable because the proposed routing uses more bandwidth than is available on the line between nodes 1 and 5. Arrangement (b) is realizable because the capacity of the lines on the ring is not exhausted. Ring demand loading is an extremely important issue which must be addressed in the design of SHR networks as it directly impacts the amount of resources required to implement a design.

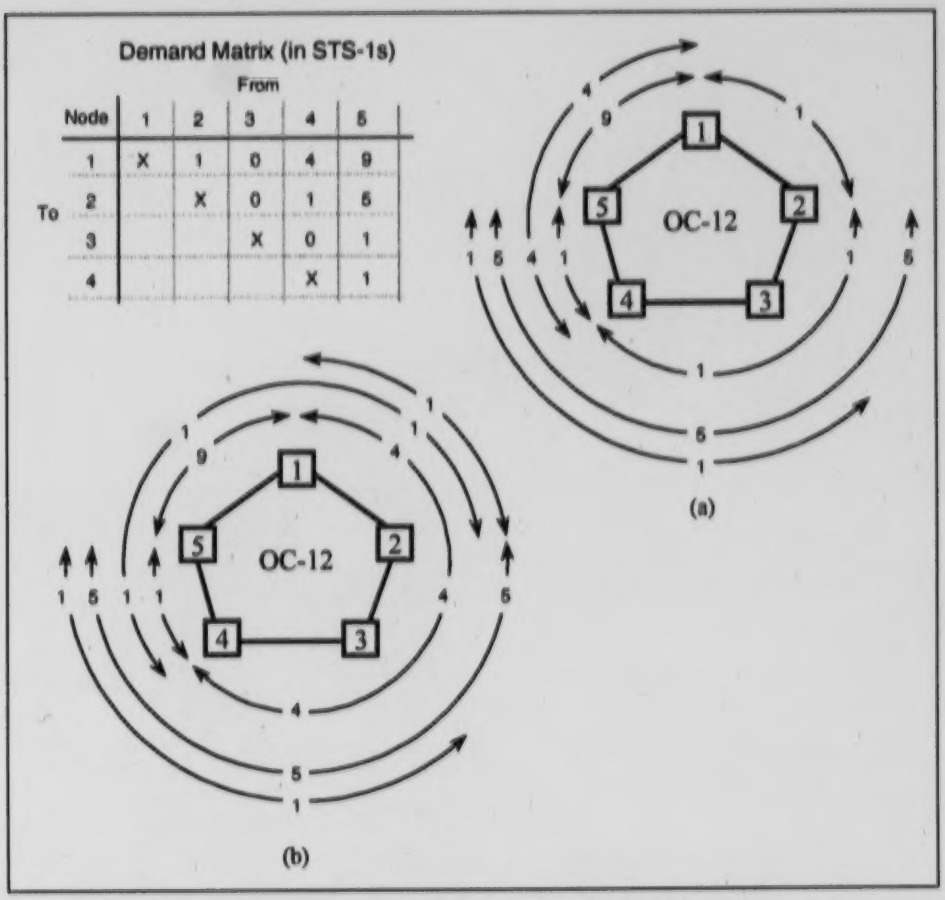


Figure 12: An example of BLSR ring loading (adapted from [4])

2.4.2 UPSR System Capacity

As with BLSRs, UPSR system capacity is described in terms of the optical line rate of the transmission system as well as the demand management quantity. Unlike, BLSRs, there is no span reuse capability with UPSRs because time slots are assigned on all lines around the ring for each demand routed on the ring. A UPSR ring loading example is shown in Figure 13. In case (a), the demand from node 1 to 4 is loaded, resulting in use of 10 of the 12 available STS-1 units of capacity around the ring. This ring is called a 2UPSR12. No further demands can be loaded because there is insufficient remaining capacity. In case (b), the demands are chosen in a different order, and the ring capacity is completely used. As with BLSRs the selection order of the demands to load onto a ring is

critical to the efficient utilization of the ring.

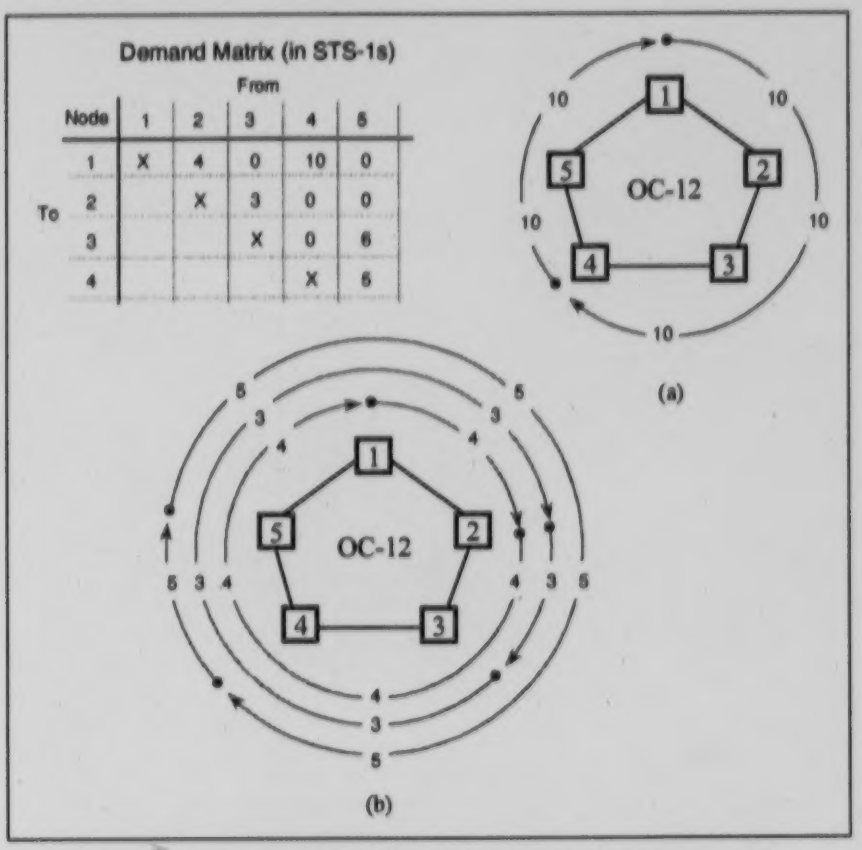


Figure 13: An example of UPSR loading (adapted from [4])

2.4.3 Add-Drop Multiplexer Considerations

In a SONET ring, Add Drop Multiplexers serve as the line terminating equipment. An *Add Drop Multiplexer (ADM)* provides the gateway for signals to enter and leave the SHR. ADMs have optical interfaces for connecting to other ADMs around the ring via the line segments which they terminate. They also provide electrical and/or optical interfaces at the node through which signals enter or leave the ring. Demands are added or dropped at one or more demand management quantities at the node. ADMs also contain switching functionality for the assignment of line time slots for added demands. This functionality is called *Time Slot Assignment (TSA)*. Some ADMs have the additional ability to reassign the time slots of demands passed through that node. This functionality is called *Time Slot*

Interchange (TSI). The ADM looks like section terminating equipment rather than line terminating equipment to such demands.

On occasion, ADMs do not have the capability to add or drop all of the line bandwidth at any one node. This is generally due to the physical restriction of limited shelf equipment space allocated to add/drop cards. For instance, an OC-12 UPSR has the line capacity of 336 DS1s, but may only be able to add or drop one half of them or 168 DS1s at any one node. Thus, one must consider the line rate, the demand management quantity, and any node add/drop limitations of the ADMs, when loading a ring system with demands.

2.4.4 Summary

Table 1 summarizes the main characteristics of the 8 theoretical types of self-healing ring. Two of the ring types have notable characteristics:

1. Line switched bidirectional rings have the capability to reuse working traffic time slots on a per span basis. This bandwidth reuse feature improves the overall demand-serving ability of the installed bandwidth on the ring. In contrast, all of the other ring types assign a demand to a time slot for all spans of the ring.
2. 4-fibre bidirectional line switched rings have an additional span failure fault recovery mechanism. A span localized switch to the protection bandwidth can be initiated in the case of a partial span failure in which only the working fibre is affected.

Table 1: Summary of Operating Characteristics for Self-Healing Rings (adapted from [7])

LINE/ PATH	UNI/BI-DIRECT.	2/4 FIBRE	Line Payload Capacity (Relative to Line Rate)	Bandwidth Reuse	Protection Switching	SONET Standard
LINE	UNI	2	1X	no	head/tail	
LINE	UNI	4	2X	no	head/tail	
LINE	BI	2	0.5X	yes	head/tail	✔
LINE	BI	4	1X	yes	head/tail, span	✔
PATH	UNI	2	1X	no	tail	✔
PATH	UNI	4	2X	no	tail	
PATH	BI	2	1X	no	tail	
PATH	BI	4	2X	no	tail	

The unidirectional path switched ring is often used because of the simplicity of operation and provisioning. Bidirectional line switched rings are preferred when maximum payload capacity is required.

2.5 Conclusion

Of the eight different types of self-healing rings, only the 2- and 4-fibre BLSR and 2-fibre UPSR subtypes are used in practice today; they have been standardized in the SONET Generic Recommendations [2,5]. The basic operation of each of the main ring types has been described. The operation and layout of one ring using any of the self-healing ring types is straight-forward. If a network could be designed such that a single ring could transport all of the payload demand, the design process would be quite easy. However, practical network designs cannot be accomplished with a single ring transport system. The resulting multiple ring design problem is a multifaceted one, having many associated network, system and technology issues. These issues are the topic of the following chapter.

3. Networks of Multiple SHRs: Design Principles and Issues

Numerous issues at the network topology, ring topology, demand routing, system technology, and design levels must be considered in order to synthesize efficient designs of multiple SHR networks. This chapter discusses aspects of each of these issues, and introduces all of the concepts which are addressed by the RingBuilder™ framework in Chapter 4.

3.1 Network Topology Considerations

Figure 14 outlines aspects of the network topology which must be considered by a planner when designing a multiple SHR network. Each aspect represents degrees of freedom that must be addressed as the planner attempts to synthesize a cost-effective solution.

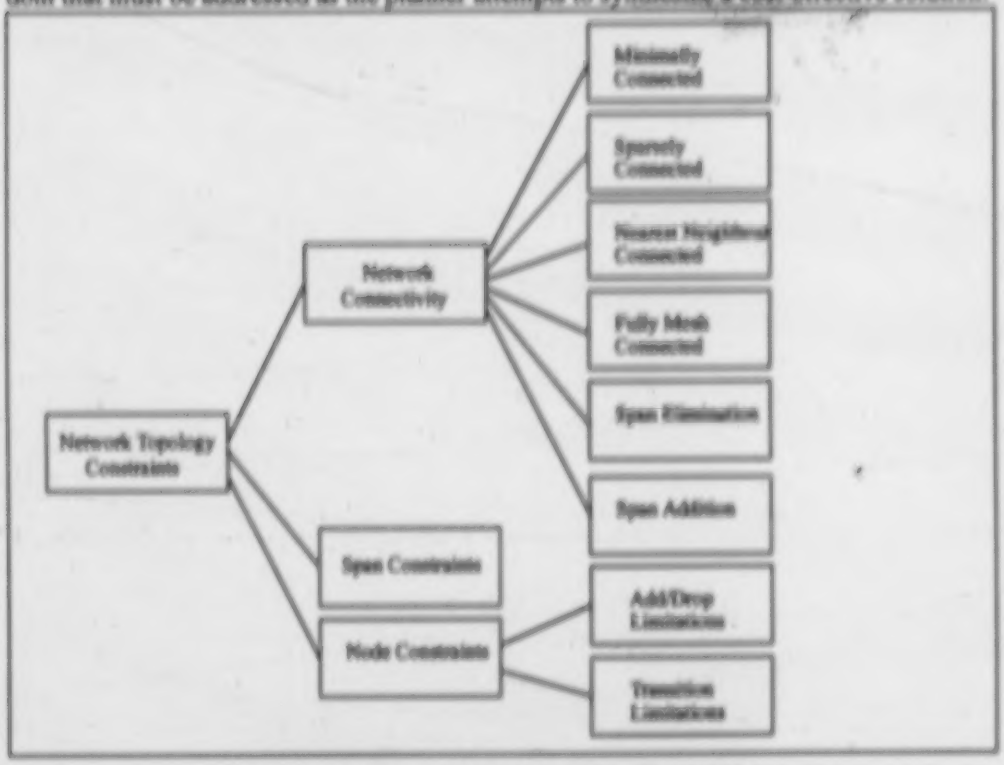


Figure 14: Network Topology Constraint Map

3.1.1 Network Connectivity

Network connectivity describes the manner in which each node is connected to the

other nodes in the network. To synthesize a ring-based network design, it is necessary for every node to have at least two spans connected to it, i.e. to be at least *2-connected*. The number of spans connected to a node is its *nodal degree*. The higher the average nodal degree of a network, the larger its ultimate solution space, as the number of potential cycles from which to choose ring topologies increases.

Often, the network connectivity is fixed before SHR network design is undertaken. Sometimes, however, the planner may modify the existing network connectivity by specifying additional inter-node connections (*span addition*), or by disusing existing inter-node connections (*span elimination*). In rare cases, the planner may be able to specify the entire inter-node connectivity structure. This is called a *green-field* design because of the freedom from existing infrastructure encumbrances. In undertaking a green-field network design, a planner may investigate several network architectures that could be used to connect the designated node sites. Suitable network architectures include minimally connected, sparsely connected, nearest-neighbor connected, or fully mesh connected.

If an entire network is 2-connected, it is called a *minimally connected* network. If a network includes some 2-connected nodes, but also some nodes of a degree higher than two, it is said to be *sparsely connected*. Most networks for which ring system designs are completed are sparsely connected. Their nodes are at least degree 2, but are often higher because of connections that exist as a result of pre-existing point-to-point systems.

A *nearest-neighbor* connected network has all nodes connected to their physically nearest neighbors. In the limit, a nearest-neighbor connected network is minimally connected (2-connected), but may be 'sparsely connected', or of even higher degree, depending on the proximity of the nodes to one another. Metropolitan networks are often 'nearest neighbor' connected.

A *fully mesh* connected network has all nodes connected to every other node. Fully mesh connected networks have the property that all demands may be routed to any destination via single-hop paths. Very few networks are fully mesh connected because of the impracticality of having direct spans between all network nodes.

Network connectivity plays a pivotal role in the size of the available solution space from which to synthesize a design. Span elimination or addition may be used by the planner to change an existing network from one network type to another. A green field network

design experiment, showing the relative attributes of each of these network types in an SHR design environment, is described in Chapter 5.

3.1.2 Node and Span Constraints

Each network is comprised of nodes and spans. These elements may have physical constraints which limit the design options available to the planner when synthesizing an SHR design. Although not all SHR design techniques consider node and span constraints, RingBuilder™ does, and thus a brief introduction to these issues follows.

3.1.2.1 Node Constraints

Node-oriented constraints include add/drop restrictions and transition restrictions. Since each node can have different add/drop and transition restrictions, these constraints must be addressed on a node-by-node basis. A node that cannot add or drop signals is said to be *add/drop restricted*. An add/drop restricted node does not have any signals entering or leaving the network, due to the lack of available equipment or physical space to add equipment. If a set of demands enters or leaves the network at an add/drop restricted node, the planner must determine whether the demand set is in error or whether the node facilities must be upgraded to remove the restriction, so that ring ADMs may be installed.

A *transition-constrained* node does not support the passing of signals from one ring system to another. Ring system ADMs are interconnected to one another via electrical patch panels and/or digital crossconnect systems. An *electrical patch panel* allows electrical signals to be manually rerouted from a source (for instance an ADM drop port on one ring) to a destination (for instance an ADM add port on a second ring). A *digital crossconnect system* (DCS) is an electronic version of the electrical patch panel. A DCS may be remotely controlled to connect many source signals from ring system ADMs to many different destination ring ADMs. A DCS may also be programmed to automatically reconnect the signals. If no electrical patch panel or DCS is available, or if existing patch panels or DCSs are fully utilized, it may not be possible to pass signals between SHR systems at a node. In this case the planner must prevent demands from crossing between ring systems at this node.

Both of these node-oriented constraints are extremely difficult to address after a net-

work design has been generated.

3.1.2.2 Span Constraints

The number of physical fibres available along a network span is a *span constraint*. There are no span constraints in the case of a green-field design, as the planner can specify the number of fibres to deploy in order to accommodate the SHR design target. However, if it is necessary to deploy an SHR design in an existing network, the number of fibres available in each span is a critical consideration. To be realizable, the resulting design must not require more fibres than exist already.

3.2 Working Path Placement Considerations

An important step in the SHR network synthesis process is the placement of the working paths on which the point-to-point demands are carried. The working path placement process comprises two key functions: demand bundling and demand routing. Figure 15 outlines the working path placement considerations.

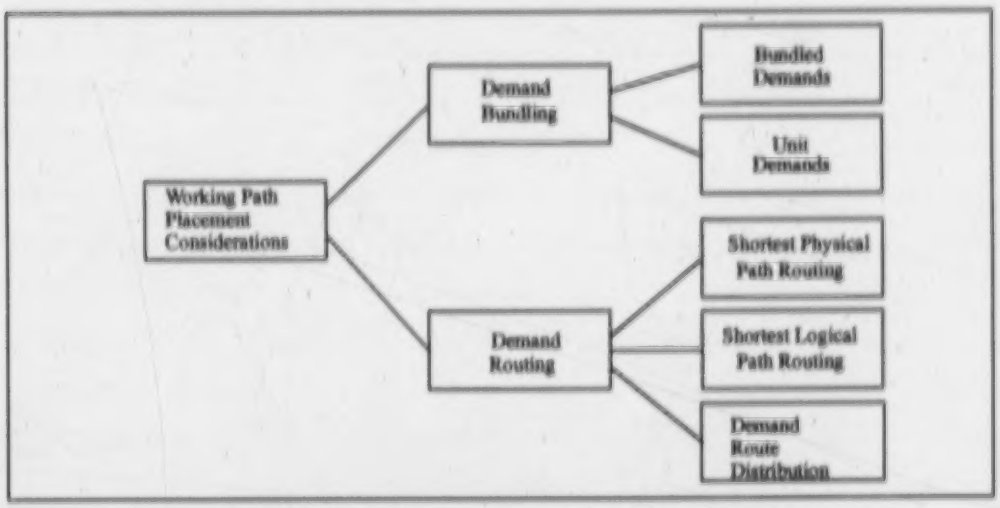


Figure 15: Working Path Placement Considerations

3.2.1 Demand Bundling

The synthesis process may treat the point-to-point demands carried by an SHR network as bundled demands or as sets of unit demands. *Bundled demands* are entities that

have to be considered as unified atomic quantities. *Atomic quantities* cannot be subdivided and processed as multiple entities. Bundled demands cannot be split across multiple parallel ring systems, and they must be routed on a single path within the ring systems they travel on. If a point-to-point demand is treated as a *set of unit demands*, each of the unit demands may be routed on different paths and different ring systems, and on different paths within each of the systems. The *unit* of the unit demand is the demand management unit (e.g. DS1, STS1, DS3) for the ring system(s) carrying the demand from source to destination.

3.2.2 Demand Routing

An efficient, realizable design is critically dependent on the routing of point-to-point demands. The routing process can take place before, during, or after *ring placement*, the selection of ring systems to deploy in an SHR design. Often, shortest path routing is used. The technique of *shortest path routing minimizes* either the total hop count or the total physical distance a demand has to travel from source to destination. The *hop count* is a count of the number of spans on which the demand travels from source to destination. Routing using hop count minimization is sometimes called *shortest logical path routing*, whereas routing using physical distance minimization is sometimes called *shortest physical path routing*. Shortest path routing attempts to minimize the amount of installed capacity in the final design. If demands are routed prior to ring selection, the use of the shortest paths possible is ensured. However, this may compromise the overall fill of the installed systems. If demands are routed during ring placement, ring fill can be maximized. However, this can potentially be detrimental to the overall efficiency of the design because the routes chosen may not be shortest paths. If demands are routed after ring selection, difficulties can still occur. The rings chosen may not be able to accommodate all of the demand presented to them, yet may still have excess capacity.

Shortest physical path routing is used when the span costs dominate over the node costs in a network, such as in an inter-city network. Shortest logical path routing is used when the node costs dominate over the span costs, such as in a metropolitan network. Node and span costs in the SHR design context are discussed in detail in Section 3.5, Existing Approaches to the Design Problem.

Demand route distribution is a technique where demands are split equally over multiple equivalent shortest path routes. This is called *k-way routing* because the demands are split over 'k' equivalent routes. In contrast, routing a demand over a single shortest path route (even if there are other possible equally short routes) is called *1-way* routing. The distribution of the demand over more routes via k-way routing can raise the overall network availability, and can result in lower installed bandwidth in a total design. 1-way routing has the advantage of ensuring that all components of a demand follow the same path through the network. This can be more efficient operationally to a telecommunications operator than k-way routing of demands, because the record keeping of the ring systems involved with any one demand is simplified.

If the demands are to be routed on a single path, but can be split among many systems, 1-way routing is dictated. If the demands are to be routed on a single path and cannot be split across systems, 1-way routing and demand bundling is dictated. If the routes can be split across multiple equivalent paths, then k-way routing may be used. The demand bundling/routing options are summarized in Table 2 . The only options that are not applicable are the ones with bundled demands and 'k' shortest path routing. *Bundled demands* are demands that consist of one or more demand management units of bandwidth that must be treated as a single entity. Because bundled demands are atomic quantities, only one route for a demand can be used even if there are 'k' shortest paths available and thus this combination is equivalent to bundled demand/single shortest path routing.

Table 2: Demand Bundling/Routing Options

Demand Routing	Demand Bundling	
	Not-Bundled	Bundled
Not shortest path routed	yes	yes
Single shortest Physical path routed	yes	yes
Single shortest Hop Count path routed	yes	yes
'k' shortest Physical path routed	yes	yes
'k' shortest Hop Count path routed	yes	equivalent to single shortest path routed
'k' shortest Physical path routed	yes	equivalent to single shortest path routed
Aggregating Routing	yes	yes
Hubbed Routing	yes	yes

3.3 System Technology Considerations

The characteristics of the network elements used in the deployment of SONET rings for the target network design are important. The physical limitations of the ADMs comprising the rings and the modularity of the system under consideration are the two main system technology issues that must be addressed by the planner. System technology considerations are shown in Figure 16.

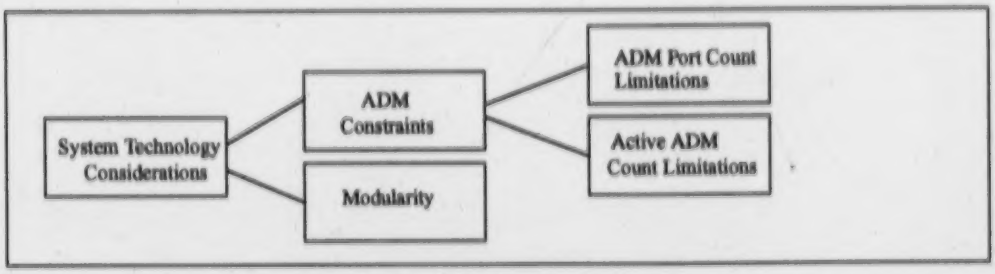


Figure 16: System Technology Considerations

3.3.1 Modularity

Modularity is the discrete capacity of a particular system in terms of the basic demand management unit. The *demand management unit* is the smallest unit of bandwidth which a system processes as a unique entity. For example, a 4-fibre OC-48 BLSR could have a modularity of 48 STS1 rate tributaries if the ring operated with STS1 as the demand management unit, or 1344 DS1 tributaries if it used DS1 as the demand management unit. A *tributary* is an add/drop signal from an ADM, and the tributary bandwidth is an integer multiple of the demand management unit of the ADM. The system design may be approached by considering a single basic demand management unit, or by considering multiple modularities so that individual demands can be handled using their native demand management quantities. RingBuilder™, to be discussed in Chapter 4, uses the former guideline.

3.3.2 ADM Constraints

Two ADM constraints must be considered in the synthesis of an SHR design. These are the maximum number of add-drop ports allowed per ADM and the maximum number of active ADMs allowed in the system.

3.3.2.1 ADM Port Count Limits

The *ADM port count limit* constraint is the maximum number of add/drop ports that can be used to source, terminate, or transit traffic on a ring. The add/drop ports are the entry and exit points for a demand on a ring. Only a fraction of the optical line bandwidth may be addressed by the ADM ports at any one ADM, due to physical space limitations or operating constraints. If a system can add, drop, or transit all of its bandwidth at a single node, it is said to be a *non-access-limited system*. If a ring system is comprised of ADMs that can only add/drop or transit a portion of the optical line bandwidth, that ring system is said to be an *access-limited system*. In Chapter 5, SHR design experiments are carried out with non-access limited systems as well as with access-limited systems where only one half of the optical line bandwidth can be accessed at any one node.

3.3.2.2 Active ADM Count

The number of active nodes in a ring system is generally limited. For instance, the SONET standard specifies that BLSRs may have up to 16 active nodes, since the APS message channel consists of 2 bytes (K1, K2), with a nibble in each containing ADM address information. Equipment manufacturers produce systems with active node limits below, at, and above this limit. In order for a system to be realizable, the active node count limitations for the equipment under consideration must be accounted for at design time.

3.4 Design Considerations

In addition to considering the network, system, and demand requirements, the designer must take into account several structural aspects of the SHR network design. These aspects include whether to consider more than one ring system technology type, what optimization strategy to pursue, what pre-load route sorting strategy to implement, what route loading strategy to use, and where to place active nodes. The *ring system technology type* is described by the ring system's optical line rate, its native demand management unit, its protection switch type (i.e. line switched or path switched), and its fibre count (2 or 4 fibre). The design considerations are shown in Figure 17.

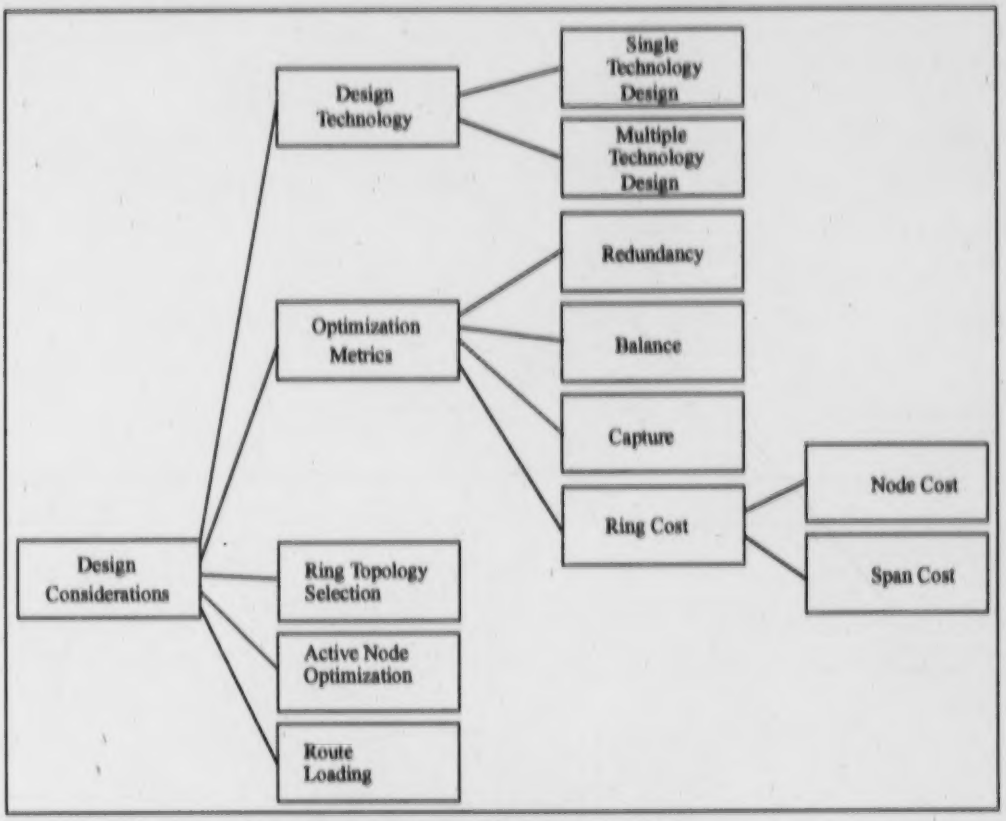


Figure 17: Design Considerations

3.4.1 Design Technology

A network design may be created by using a *single technology* (e.g. 2-fibre OC-12 UPSR-only or 4-fibre OC-48 BLSR-only) or by choosing from multiple technologies on a ring-by-ring basis. Although the use of only one technology is simpler, an inefficient design may result, especially if the system component capacity is not well-matched to the demand cross-sections. Choosing from multiple technologies for each ring is more complex, but may ultimately result in a more efficient design.

To generate an efficient single-technology design, the technology's payload capacity must be matched to the size of the demands to be carried by the network. If the system capacity is too large for the majority of the demand cross-sections, the systems deployed will have low fill and thus excess capacity will be deployed. If the system capacity is small

relative to the demand cross-section, then multiple 'stacked' rings can occur in the network design. Stacked rings are inefficient if a single larger capacity ring system can take the place of the stacked systems at a lower overall cost. However, operational efficiencies are realized when a single system type is used to generate the network. They include lower inventory counts (hence lower costs for equipment spares), potentially lower training costs for craft personnel, and simplified network operations activities, due to the common architecture of all the network elements.

A multiple technology design can be generated by various methods. The techniques presented here as examples can be compared to RingBuilder™ (described in Chapter 4) by the reader. First, a single technology design can be generated as an exploratory exercise, to better characterize the network and the demand payload it is slated to carry. This can evolve to a multiple technology SHR network design by manually identifying instances where multiple lower capacity ring systems can be replaced by a single higher capacity ring system. A second approach involves the examination of each available system technology for each ring choice. This is labor-intensive as each potential ring must be constructed with each system technology. A third method involves the creation of rings using the largest capacity technology until the system fill drops to a point where a smaller capacity technology is more viable, continuing with that technology until it is less viable than the next smaller technology, and repeating the 'downsizing' process until the design is complete. This approach is simpler than the second approach as only one technology is examined per ring choice until the fill drops sufficiently, but the determination of the fill thresholds at which to change technologies is difficult. The technology 'thresholds' are 'learned' values, however, and once determined can be used on subsequent designs. RingBuilder™ can synthesize both single- and multiple- ring network designs. RingBuilder™ evaluates all available technologies for every ring selected in the synthesis process.

3.4.2 Optimization Metrics

The planner's main goal in designing an SHR network is to minimize total cost while ensuring that all demands are carried. To minimize total design cost, the individual rings chosen must be the most cost effective. To ensure that the rings are cost effective, the installed capacity available on the rings must be maximally utilized, and all excess costs

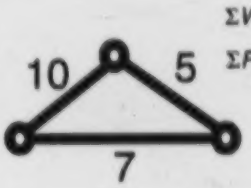
must be minimized. Capacity utilization is measured by examining how well the demands carried by individual rings utilize the available bandwidth. Excess costs in a ring design result from having too many ring systems, and from demands crossing between too many individual rings as they travel from source to destination in the network. This section describes several fundamental properties of rings that can be used in an SHR design cost optimization framework: redundancy, balance, capture, and cost.

3.4.2.1 Redundancy

Redundancy is a measure of the amount of protection capacity that has to be deployed relative to the used working capacity present in a system or network design. Redundancy is used to evaluate the bandwidth efficiency of survivable network designs, and can be used to evaluate mesh, point-to-point, and ring survivable network designs. All survivable rings, irrespective of type, have redundancy of at least 100%. Redundancy can be expressed in three ways: simple redundancy, geographical redundancy, and modular geographical redundancy. *Simple redundancy* is the sum of the protection capacity installed divided by the protected used working capacity [9]:

$$redundancy = \left(\sum_{i=0}^{n-1} P_i\right) \Big/ \left(\sum_{i=0}^{n-1} W_i\right) \tag{4}$$

where n is the number of spans in the cycle, P_i is the protection bandwidth on span i, and W_i is the working bandwidth on span i. The required protection bandwidth on each span for a BLSR is the peak of the working bandwidth carried on the ring. For a UPSR, the required protection bandwidth on each span is the sum of the working bandwidth carried on the ring. A BLSR example is shown in Figure 18.



$\Sigma W_i = 10 + 5 + 7 = 22$

$\Sigma P_i = 10 + 10 + 10 = 30$

$$redundancy = \frac{\sum P_i}{\sum W_i} = \frac{30}{22} = 1.36$$

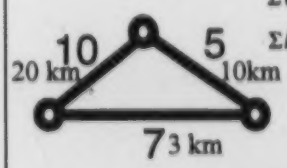
Figure 18: Ring Capacity Redundancy Measurement

Geographic redundancy is a modification of simple redundancy that weights the working capacity and the protection capacity by the geographical distance that each respectively travels. Geographic redundancy is a more suitable metric than simple redundancy in networks where there is high variance in the lengths of the spans that make up the network and where distance related costs are significant relative to terminal costs. Geographic redundancy can be expressed as:

$$geographic\ redundancy = \left(\sum_{i=0}^{n-1} P_i \cdot d_i\right) \Big/ \left(\sum_{i=0}^{n-1} W_i \cdot d_i\right) \quad (5)$$

where n is the number of spans in the cycle, P_i is the amount of protection bandwidth on span i, W_i is the amount of working bandwidth on span i, and d_i is the physical length of span i.

An example of a geographic redundancy calculation for a BLSR is shown in Figure 19.



$\Sigma W_i d_i = 10*20 + 5*10 + 7*3 = 271$

$\Sigma P_i d_i = 10*20 + 10*10 + 10*3 = 330$

$$geographic\ redundancy = \frac{\sum P_i d_i}{\sum W_i d_i} = \frac{330}{271} = 1.22$$

Figure 19: Geographic Redundancy Measurement

Modular geographic redundancy is a modification of geographic redundancy that

considers any unused working capacity in the modular system. Modular geographic redundancy is expressed as the sum of the protection bandwidth distance products and the margin distance products divided by the sum of the working bandwidth distance products. *Margin* is defined as the unused working capacity. Modular geographic redundancy provides the most informative figure of merit when evaluating different network designs by simultaneously taking into account system utilization and installed bandwidth. Modular geographic redundancy can be expressed as:

$$\textit{Modular geographic redundancy} = \left(\sum_{i=0}^{n-1} P_i \cdot d_i + \sum_{i=0}^{n-1} M_i \cdot d_i\right) \Big/ \left(\sum_{i=0}^{n-1} W_i \cdot d_i\right) \qquad (6)$$

where n is the number of spans in the cycle, P_i is the protection bandwidth on span i, M_i is the margin bandwidth on span i, W_i is the amount of working bandwidth on span i, and d_i is the physical length of span i. An example of the modular geographic redundancy calculation for a BLSR is shown in Figure 20.

$\Sigma W_i d_i = 10*20 + 5*10 + 7*3 = 271$
$\Sigma P_i d_i = 10*20 + 10*10 + 10*3 = 330$
$\Sigma M_i d_i = (12-10)*20 + (12-5)*10 + (12-7)*3 = 125$

$$\textit{Modular Geographic Redundancy} = \frac{\sum P_i d_i + \sum M_i d_i}{\sum W_i d_i} = \frac{455}{271} = 1.68$$

System Modularity 12 Units/span

Figure 20: Modular Geographic Redundancy Measurement

3.4.2.2 Balance

Balance is a measure of the working capacity carried by each ring span relative to the amount of protection capacity embedded in the ring. It is inversely proportional to redundancy. Like redundancy, it can be expressed as simple balance, distance weighted balance, or modular distance weighted balance. The balance figure of merit has a dynamic range of 0.0 - 1.0, where 1.0 is perfect balance with complete utilization of all bandwidth on all spans in the ring. A balance of 1.0 corresponds to the minimum of 100% redundancy

in the ring under consideration. Balance is sometimes referred to as *balance efficiency*, because of the range of values it encompasses. In the case of modular rings, balance can also be thought of as a normalized measure of ring *fill* (working capacity utilization). Higher balance efficiency is reflected in the system cost by a reduction in the total amount of fibre, ADMs, repeaters, and lower DCS costs.

Balance efficiency can be expressed as:

$$\textit{balance efficiency} = \left(\sum_{i=0}^{n-1} W_i\right) \Big/ \left(\sum_{i=0}^{n-1} P_i\right) \tag{7}$$

In the example shown in Figure 18, the BLSR ring has a ring balance efficiency of 22/30 = 0.735.

Distance-weighted balance is the inverse of distance-weighted redundancy. It can be expressed as:

$$\textit{distance-weighted balance} = \left(\sum_{i=0}^{n-1} W_i \cdot d_i\right) \Big/ \left(\sum_{i=0}^{n-1} P_i \cdot d_i\right) \tag{8}$$

The distance-weighted balance for the example shown in Figure 19 is 271/330 = 0.821.

Modularized distance-weighted balance is the inverse of modularized distance weighted redundancy. It can be expressed as:

$$\textit{modularized distance-weighted balance} = \left(\sum_{i=0}^{n-1} W_i \cdot d_i\right) \Big/ \left(\sum_{i=0}^{n-1} P_i \cdot d_i + \sum_{i=0}^{n-1} M_i \cdot d_i\right) \tag{9}$$

The modularized distance-weighted balance of the example shown in Figure 20 is 271/455 = 0.596.

RingBuilder™ makes use of modularized distance-weighted balance. This is illustrated in Figure 21.

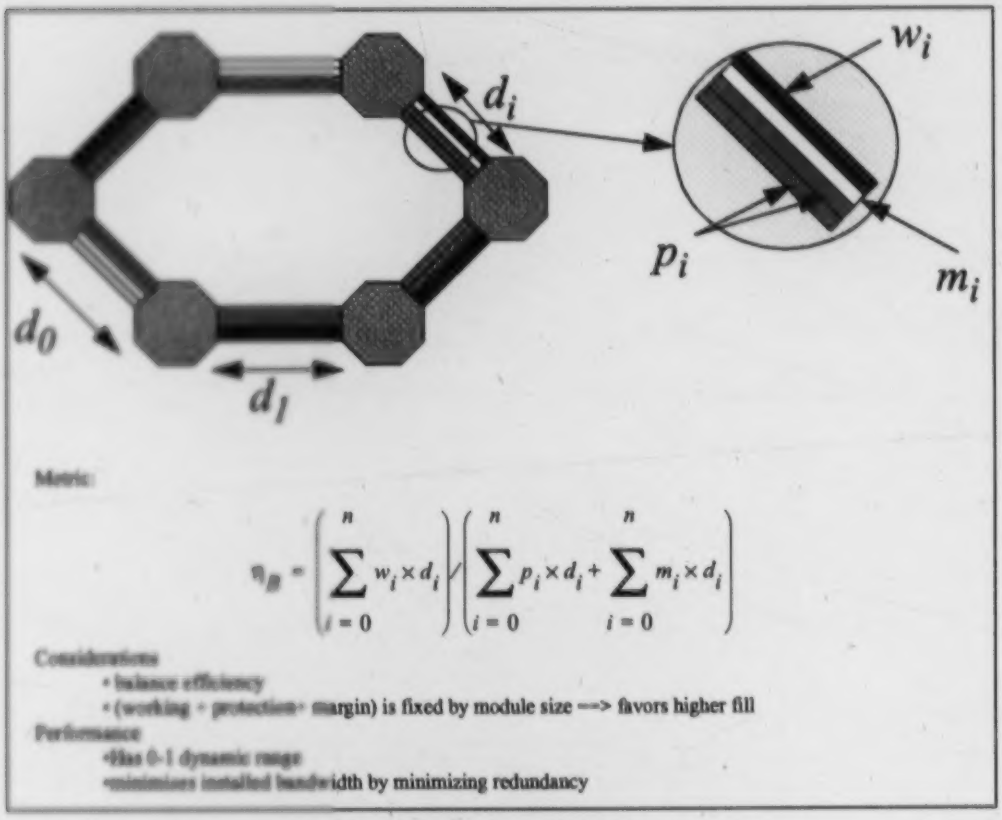


Figure 21: Balance Efficiency

3.4.2.3 Capture

From [9]:

"The efficiency of a candidate ring in terms of minimizing interface costs due to transitions of demands from ring to ring is defined by its traffic capture efficiency"

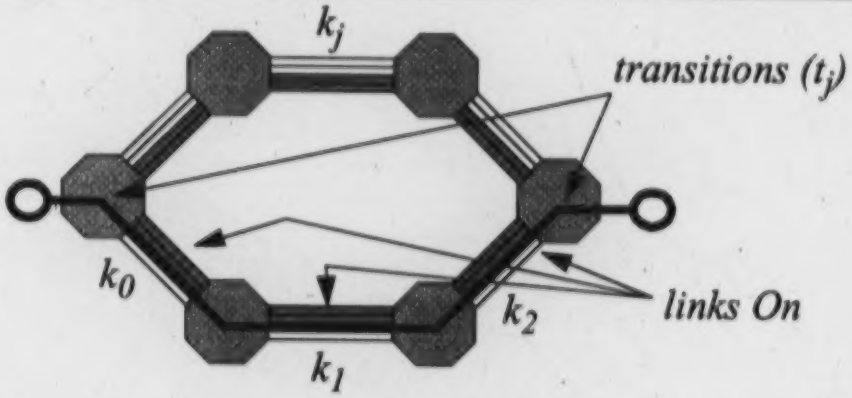
Traffic capture efficiency, or *capture* can be expressed as:

$$capture = \frac{\left(2 \cdot \sum_{i=0}^{n-1} l_i\right) - \sum_{j=0}^{n-1} t_j}{\left(2 \cdot \sum_{i=0}^{n-1} l_i\right)}, \qquad (10)$$

where n is the number of spans in the ring, i is the span number, j is the node number, l_i is

the number of links on that span, t_j is the number of transitions at node j. A *link* is a demand management unit of transport bandwidth within a span of the ring under consideration.

Capture is a normalized metric with a range of 0.0 - 1.0. If all demands that a ring carries are fully contained within the ring from source to destination, that ring is said to have perfect capture, and the capture figure of merit is 1. Conversely, if all of the demands carried by the ring come onto the ring, travel for a single hop and then leave the ring once more, the capture is 0. This 'worst case' capture scenario exhibits the maximum number of transitions (2) per demand management unit of bandwidth carried by each span. Capture does not have any modularity or distance weighting associated with it, since it focuses on the cost elements in the nodes of the network. Higher capture efficiency is reflected in a design as reduced DCS cost, as the number of rings required to transport the demands from source to destination is minimized. Capture efficiency is summarized pictorially in Figure 22.



- Metric:

$$\eta_C = \left(2 \times \sum_{j=0}^{n} k_j - \sum_{j=0}^{n} t_j\right) / 2 \times \sum_{j=0}^{n} k_j$$

- Considerations
  - This metric looks for rings with minimum transition counts
  - η_C is a normalized transition count
  - linksOn emphasizes distance of demand route travel within cycle
- Performance
  - Has desired 0 - 1 dynamic range
  - Minimizes number of ring-to-ring transits for demand routes

Figure 22: Capture Efficiency

In the example shown in Figure 23, demand 'm' is completely captured by ring Q, and thus no excess access costs are generated in its transport.

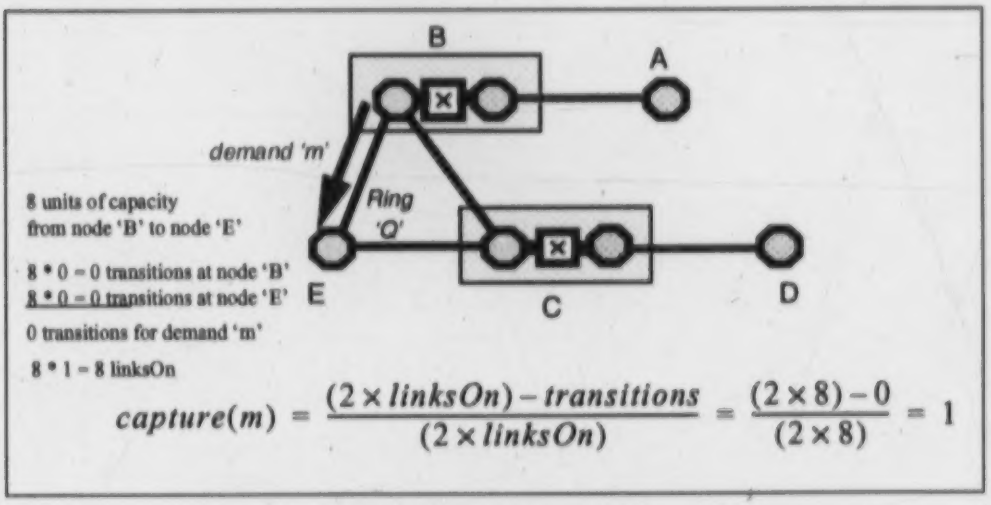

B
A
demand 'm'
Ring 'Q'
8 units of capacity
from node 'B' to node 'E'
8 * 0 = 0 transitions at node 'B'
8 * 0 = 0 transitions at node 'E'
0 transitions for demand 'm'
8 * 1 = 8 linksOn
E
C
D
$capture(m) = \frac{(2 \times linksOn) - transitions}{(2 \times linksOn)} = \frac{(2 \times 8) - 0}{(2 \times 8)} = 1$

Figure 23: Ring Traffic Capture –Maximum Capture

In Figure 24, demand 'n' is not completely captured. It is said to be minimally captured, since extra interface costs are generated in interfacing ring Q to other rings in the design to enable the demand to travel from its source to its destination.

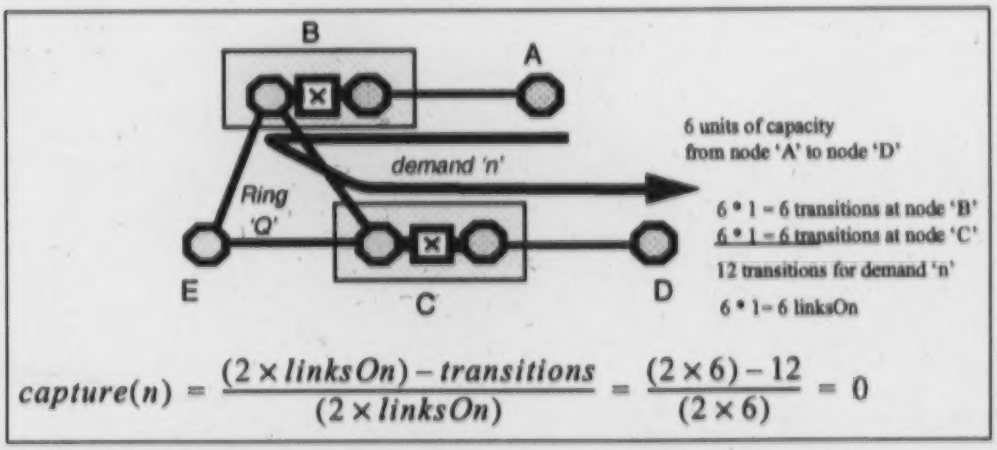

B
A
6 units of capacity
from node 'A' to node 'D'
demand 'n'
Ring 'Q'
6 * 1 = 6 transitions at node 'B'
6 * 1 = 6 transitions at node 'C'
12 transitions for demand 'n'
6 * 1= 6 linksOn
E
C
D
$capture(n) = \frac{(2 \times linksOn) - transitions}{(2 \times linksOn)} = \frac{(2 \times 6) - 12}{(2 \times 6)} = 0$

Figure 24: Ring Traffic Capture – Minimum Capture

In Figure 25, an intermediate case is shown. Demand 'p' is partially captured by ring 'Q' because the demand starts on ring node 'E', but transits off the ring at node 'C' to go to its destination at node 'D'. The capture figure of merit is 0.5.

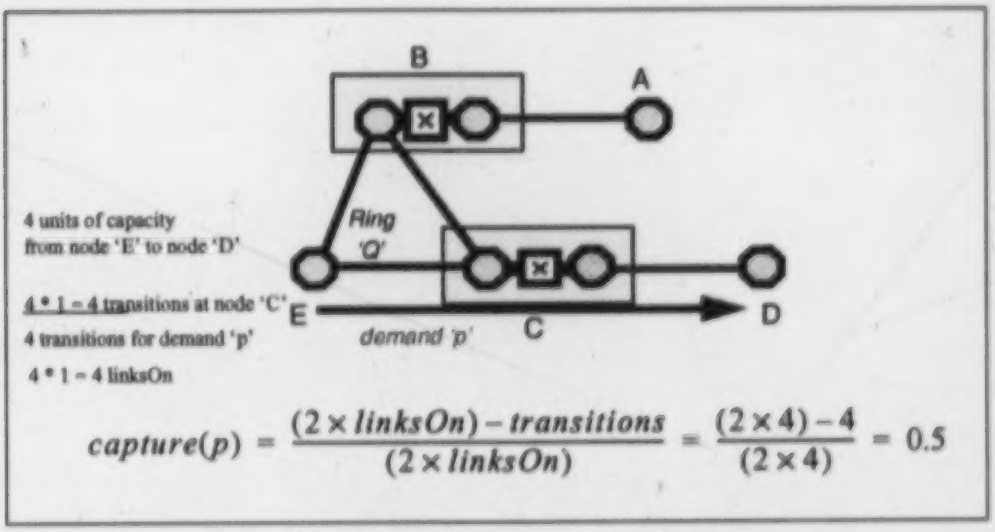

B
A
4 units of capacity
from node 'E' to node 'D'
Ring
'Q'
4 * 1 = 4 transitions at node 'C'
4 transitions for demand 'p'
4 * 1 = 4 linksOn
E
demand 'p'
C
D
$capture(p) = \frac{(2 \times linksOn) - transitions}{(2 \times linksOn)} = \frac{(2 \times 4) - 4}{(2 \times 4)} = 0.5$

Figure 25: Ring Traffic Capture – Partial Capture

3.4.2.4 Ring Cost

The final fundamental quantity that can be used to evaluate the fitness of a ring candidate in a design is ring cost. A *ring candidate* is defined by a cycle's trajectory through the network nodes and spans, by the ADMs present at each of the active nodes, by the regenerators present at the pass-through nodes, by the fibre present in the spans, and by the set of demands that it carries. Ring cost is composed of two subcomponents: node cost and span cost. *Node cost* includes the cost of the ADM equipment placed at active nodes in a ring, and the repeaters placed at pass-through nodes. The ADM cost is comprised of the cost of the common equipment and the provisionable equipment. The *common equipment* includes ADM power supplies, equipment shelves and racks, electronic subsystems, and line optical interfaces. The *provisionable equipment* includes the tributary access cards which are provisioned as required for adding or dropping demands or for transiting demands to other rings.

Span cost includes the cost of the fibre and any inter-node regeneration equipment that exists in the line segments connecting ring nodes together.

A design technique that uses RingBuilder™ to synthesize SHR networks based on direct cost optimization of the ring node and span costs is introduced in the next chapter.

3.4.3 Ring Topology Selection

Another fundamental design consideration is the choice of ring topology for each of the constituent rings in an SHR design. *Ring topology* is defined as the ring's path through the nodes and spans of the network. It is related to the graph theoretic concept of the cycle in that the path that a ring traces through a collection of nodes is a cycle in the graph representation of the network.

Not all SHR network design techniques explicitly use cycles or determine the complete set of potential cycles in a network. RingBuilder™ was the first technique disclosed to use knowledge of the complete cycle set in the SHR network synthesis process [9].

It is not generally possible for a planner to exhaustively determine every potential distinct cycle in a network, or to examine each resulting ring topology. Therefore, only a portion of the potential ring topologies can be examined when ring choices are considered. A set of manually discovered cycles based on intuition and experience can be used to generate a set of ring topologies for evaluation. This approach is very useful when a planner can examine a network and see the 'natural' locations for cycles. In fact, in the manual generation of a network design, this step is implicitly performed when demands are laid out or spans are placed to interconnect the central office nodes.

Ring topology determination is just one part of the process of finding potential rings, which in turn is a part of the process of determining the set of rings with which to implement an SHR network design. The role of cycle finding in the design synthesis process is described in detail in the next chapter.

3.4.4 Active Node Optimization

Once a ring topology is chosen, *Active Node Optimization* determines which nodes to make active and which nodes to make inactive. Given a ring topology, the number of rings that can be formed by selective activation of nodes through the addition of ADMs can be expressed as:

$$numberOfRings = \sum_{n=2}^{n=m} \binom{m}{n} = 2^m - m - 1 \qquad (11)$$

where m is the number of nodes in the ring topology and n is the number of active nodes. The task of trying every combination of active nodes for potential inclusion as a ring candidate can be very time consuming on all but the smallest network designs.

3.4.5 Demand Route Loading

Demand route loading is the step where the routed demands that have to be carried by the network are mated to the rings that form the overall network design. The loading process is different for each of the ring types that can be used to form the design. The system technology, the physical system constraints, and the network span and node constraints must be taken into consideration when loading the cycle to form a ring in the network design.

While optimal techniques do exist for loading certain types of rings, heuristic techniques are often used. These heuristics, while not strictly optimal, can yield near-optimal results, and are often much less time-intensive than provably optimal approaches [4]. Because of the physical significance of capture and balance as figures of merit for evaluating rings in a design, capture- and balance- oriented ring loading heuristics may be used to optimize the design. These heuristics were conceived during RingBuilder™ development and are discussed in the next chapter.

3.5 Existing Approaches to the Design Problem

Several automated tools have been developed to assist the planner with the design of telecommunications networks using rings as elemental building blocks. Two general methodologies exist: analysis and synthesis.

Analysis tools can be as simple as spreadsheets to calculate costs, or as sophisticated as a CAD tool to generate a costed bill of materials and to automatically load rings once they have been specified by the planner. An example of a commercial analysis tool is the Nortel SONET Planner ring analysis tool.

Synthesis tools aim to generate partial or complete designs, given a network struc-

ture and a set of demands to be carried. Several approaches to the synthesis problem exist; from simple automation of the tasks intuitively performed by a planner to arrive at a design, to non-intuitive yet highly structured algorithmic techniques. A brief survey of several key approaches to design analysis and synthesis is found below.

The following discussion provides a brief overview of some of the main existing approaches to the SHR design problem. The reader is directed to 'A Comparative Survey of Methods for Automated Design of Ring-based Transport Networks', TRLabs Technical Report TR-97-04 [4] for further details of the methods described, as well as descriptions of other methods.

3.5.1 Nortel SONET Planner

Nortel, in conjunction with the Stentor group of companies, has been developing automated *analysis* tools to assist the planner in the design of a ring-based network. These tools are called VT Planner and Transition Planner [11]; the latest revision of this tool is called Nortel SONET Planner. Each of these GUI-based tools is similar in that the network architect enters a manually-generated design using experience, heuristics, intuition, and some guidelines. The network demands and the proposed ring topologies are entered into the analysis tool, which loads the ring topologies, costs out the design, generates equipment reports, and identifies any unserved demand. SONET Planner allows the network planner to develop a multiple-year strategy to evolve to a target network architecture

The generation of a design using the SONET Planner environment involves five main steps [4]. First, all network demand and topology information is entered. For a multi-year design case study, this information must be entered separately for each year of the study. SONET Planner accepts demand information in two demand management quantities: DS3 and STS3c. An *STS3c* is a 155.52 Mb/s data stream organized as a concatenated set of 3 STS1s. It is interesting to note that SONET Planner's predecessors used lower bandwidth demand management quantities. VT Planner accepted DS1 and DS3 demand management, and Transition Planner used DS3 bandwidth management. This is indicative of the trend towards higher 'granularity' of bandwidth management as the transmission rates of FOTS increases.

Once the demand and network information has been captured, SONET Planner

allows the network architect to specify the location and size of existing non-SONET transmission systems. This is a very useful feature for planners involved in the migration of existing networks to SONET rings. Several strategies for migrating from non-SONET systems to SONET rings may be utilized. For instance, a planner may specify that SONET rings are to be utilized only once all existing non-SONET capacity is utilized. An alternative strategy is to specify that all growth in demands must be handled by SONET rings. Yet another allowed option is to set a 'breakpoint' year after which all new demands have to be placed on SONET systems. A further evolution of the 'breakpoint' year is to set a period in which demands must be migrated to SONET from a non-SONET system.

Once the migration strategies have been specified, the demands are routed on existing systems. Demands can be routed manually on a per-demand basis, or can be automatically routed over shortest logical paths, starting with the largest demands and continuing until the smallest ones have been routed. This heuristic is similar to the demand routing heuristic used in RingBuilder™.

Once routing has been completed on existing systems, the planner can specify new ring systems that SONET Planner will utilize to handle the remaining unserved demand. Multiple alternative system types may be considered from a set of supported ring technologies ranging from OC3 UPSR to 4 fibre OC-192 BLSR.

The quality of the ring design depends on the experience and intuition of the planner. A strategy that can be used is to search for groups of network nodes which have significant amounts of common demand flowing between them. This identification of a community of interest can be used to identify ring topologies for constituent rings in a target SHR design.

The SONET Planner can generate many detailed equipment reports once a satisfactory target design has been found. Detailed equipment inventory reports and cost estimates are extremely useful for the planner.

Overall, SONET Planner is a very powerful tool for the experienced network planner. The ability to examine multiple migration strategies from a non-SONET transport network infrastructure, coupled with the multi-period design analysis framework, are extremely useful features. The major drawback is the reliance on the planner to determine a target topology for analysis. Typically, the planner must iterate through numerous trials to find a good result. Colleagues in industry report projects where as much as 18 months of

time was devoted to design studies using such tools for a single metropolitan ring network design. A screen capture from a version of SONET Planner is shown in Figure 26.

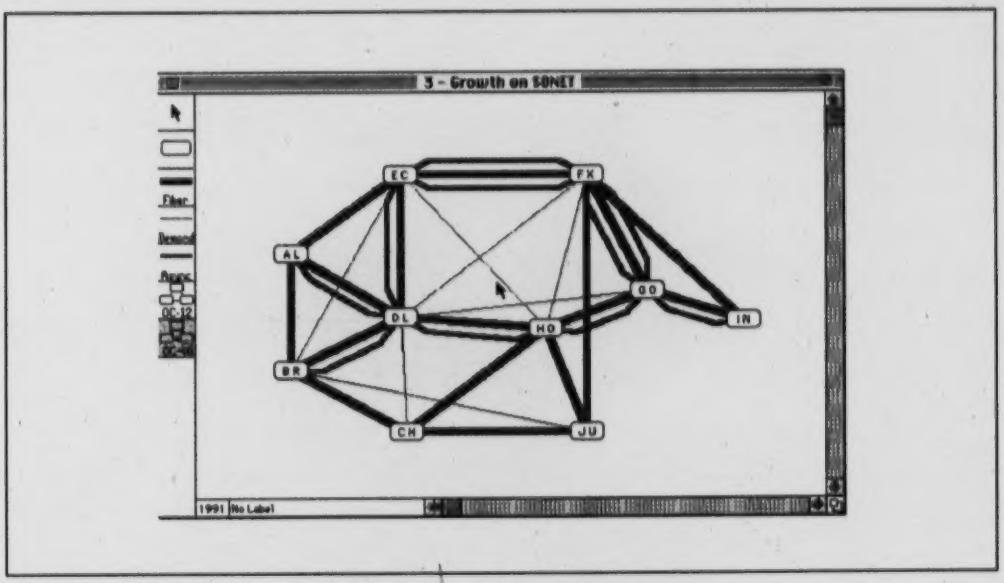


Figure 26: Nortel SONET Planner

3.5.2 Genetic Algorithms

The genetic algorithm approach is a form of optimization strategy which patterns its behavior on the process of evolution of biological systems in nature [12]. It was thought initially to show great promise as a network synthesis technique for networks of multiple self-healing rings [13,14]. As applied to the ring design problem, one starts with a 'ring covering' of the target network, and then through successive generations, evolves the covering using operators such as mutation, recombination, crossover, and reproduction on the ring elements in the network. The entire design generated in this manner must be closely examined from generation to generation to determine when a satisfactory design has been achieved.

While initial proof of concept work was carried out at TRLabs, no formal SHR design system using GAs has yet been disclosed.

3.5.3 Bellcore SONET Toolkit

Bellcore has carried out much work in the area of automated synthesis tools for network design. Their current ring design software is embedded in a generalized network design package called the SONET Toolkit, using methods developed in a suite of tools called 'Strategic Options' [15,16]. This is a multi-architecture, multi-period, synthesis system. It specifies when SHRs are to be used and when point-to-point systems are deemed to be more cost-effective. Ring selection and architecture selection are performed independently [4].

The method is divided into 4 main blocks: demand bundling, ring selection, ring topology optimization and architecture selection. The first three blocks are the 'Strategic Options' methods which synthesize ring system/ multiple diversely routed point-to-point systems. The last block is a module which compares non-ring SONET transport system alternatives to the suggested ring architecture.

The basic approach of the Bellcore methodology is to first examine the groups of nodes in the network based on their mutual coupling via point-to-point demands. Small local demand groups are bundled together by strategically interconnecting them to local hubs, thereby aggregating demand. This is the *demand bundling* process, and seems to be optimized for making metropolitan inter-switching centre routing decisions. Switch planners must often decide between making direct connections between switching centres or using an intermediate local tandem switch as a hub. Hub locations are then examined, and communities of interest involving the hubs are identified. This determines potential hubs that should reside together on single rings. Rings are constructed between these resulting groups of hubs. The *ring selection* process is an iterative heuristic, with the net goal being the lowest cost interconnection rings. The ring construction process determines the actual routes of the demands from source to destination. The method used is to concatenate chains of shortest paths connecting clustered nodes together. Once rings exist, a heuristic *topology optimizer* is invoked which attempts to find minimum topology cost realizable ring alternatives to the initial ring. The *topology costs* include fibre cost, regenerator cost, and construction cost. The system capacity is considered after the ring topology has been identified, and thus, for large demands, stacks of identical rings can exist.

This is the most mature ring network synthesis system, and a version of this software

is commercially available. Multiple SONET architectures are considered in a multi-period synthesis framework. Limitations of the method include the fact that the ring alternatives are all based on clusters of nodes grouped by communities of interest. This may cause efficient rings that contain groups of nodes that are not in single communities of interest to be ignored. Further, when installed systems reach capacity exhaust, the new systems that are added follow the existing system topologies [4]. The Bellcore approach is summarized in Figure 27.

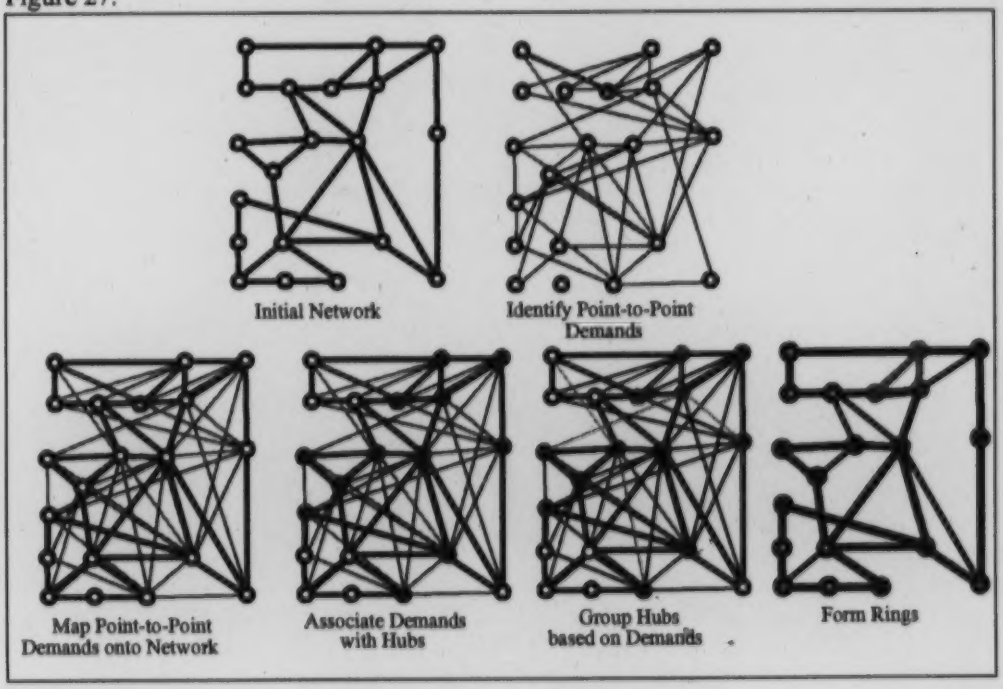


Figure 27: Bellcore SONET Toolkit

3.5.4 Hierarchical Rings

This network design approach imposes a hierarchical arrangement of rings on a network to satisfy the demand requirements [17]. The grouping of nodes at any level in this hierarchy is based on the determination of the relative community of interest between the nodes. At each level in the hierarchy, a determination is made as to which nodes should be grouped together due to local connectivity patterns. Any demand that cannot be satisfied within the ring becomes a 'foreign' demand which has to traverse a ring at the next level of the hierarchy in order to get to its destination. This method does not explicitly account for

system capacities of each ring, but that can be accommodated during the demand assignment stage. In order to get reasonably efficient results, symmetry of demand patterns at each level of the hierarchy must exist to ensure that the overall fill of the deployed rings is high. The method assumes that rings have the same, fixed, number of nodes [4]. This assumption limits the usefulness of the technique in real-world design problems, especially with non-green field designs. Further, the enforced hierarchical structure is inefficient in networks where the demand patterns are highly distributed rather than hubbed.

3.5.5 Eulerian Ring Covers

This design approach synthesizes ring network designs by forming covers of the span set of a network [18]. Routing is assumed to be completed prior to the covering stage. The network that is initially addressed must be *Eulerian*. A network is Eulerian if it is possible to have a single closed, potentially looping, path which covers every span just once. This depends on each network node having an even number of spans associated with it. If a network is not Eulerian, it can be augmented through the addition new virtual spans to make it appear Eulerian. Once an Eulerian cycle is formed, it can be reduced into a set of cycles which forms a cover. The spans added by the augmentation step do not have to physically exist, unless they remain after reduction.

This approach is intended primarily for unidirectional rings, but through heuristics, can be applied to bidirectional rings. The basic approach does not consider modularity, or glass-through nodes. It can generate coverings which require spans that do not exist in the network as presented to the synthesis system. A further weakness is that it considers covers, and so cannot leave unused any spans that are of little value to the network.

3.6 Conclusion

This chapter introduced the network-level, system-level, and design-level concepts that must be considered in the synthesis of a multiple self-healing ring network. Existing approaches to the multiple SHR design problem have been introduced, most notably two approaches in commercial form, one an analysis framework, and one a synthesis framework. The next chapter will introduce RingBuilder™ and show how it addresses the network, system and design level decisions that must be made in the SHR design problem.

4. Synthesis of SHR Networks with RingBuilder™

4.1 Introduction

The ring network design problem is an extremely complex one for human planners to address manually and, thus, considerable industry interest exists in automated techniques. Automated analysis tools like Nortel SONET Planner only address two aspects of the design problem, namely design visualization and system costing. A synthesis approach is preferred, since it additionally proposes solutions which simultaneously take into account all of the desired design objectives. The synthesis approaches published to date have been heuristic in nature. These techniques do not address all of the requirements of a methodology that can:

10. generate complete network designs without human intervention,
11. simultaneously take into account the number, sizing and placement of systems with capacity limitations,
12. cope with non-ideal topologies.

Bellcore SONET Toolkit, which is perhaps the best of the existing solutions, has the fundamental limitation of limited design space evaluation due to its reliance on the detection of communities of interest.

The new automated iterative synthesis technique described in this chapter, called RingBuilder™, addresses the desired design objectives simultaneously and produces efficient SHR network designs. RingBuilder™ is unique in that it is the first method disclosed which makes use of the complete cycle set in an iterative optimization framework. The approach is technically a greedy one (in the algorithmic computational sense) because it selects the best ring at each iteration of the design. Greedy optimization produces near-optimal rather than globally optimal solutions because the choices at each iteration are not independent.

In the course of the development of RingBuilder™, two hypotheses for ring selection were formulated: balance vs. capture optimization and specific progress direct cost optimization. Investigation of the relative performance of these techniques and of the design parameters using these techniques is the subject of Chapter 5: Experimental Results.

4.2 Basic RingBuilder™ Framework

Figure 28 outlines the basic operation of RingBuilder™. The Ringbuilder™ process starts with cycle finding, in parallel with a demand routing process. The cycle finding process attempts to find all of the distinct cycles in the network. Separately, all point-to-point demands that are to be carried by the network are routed over the available spans of the network using shortest path routing. Either the hop count or physical distance of the paths can be minimized in this step using 1-way or k-way routing. "One-way routing" routes all of the demand bundle over a single shortest path; if more than one equally shortest path is found for a given demand pair, only one of them is chosen. On the other hand, k-way routing attempts to equally utilize all shortest paths between two nodes. Demand bundling must be considered during the routing process; demands may be treated either as unit entities at the demand bandwidth management level, or as unified entities at their native bandwidth. The former treatment generally results in more efficient system utilization and higher overall connection path availability for individual payloads. The latter results in systems that are easier to administer and do not suffer from problems that occur when portions of demands are routed different ways on different systems.

Once cycle finding and demand routing have been completed, the iterative design process begins. The process iterates until either all of the demand has been transported on a set of deployed rings, or until no further demands may be accommodated on ring systems. In each iteration, each cycle is considered as the template for a prospective ring system. Each such "candidate" is loaded with demand segments on the routes of the graph which intersect the cycle. It is then is evaluated for its fitness relative to the other ring candidates under consideration. If the cycle under consideration is determined to be the best candidate yet found, it is stored, replacing the previous best candidate. After all of the cycles in the cycle set have been considered as ring templates in this manner, the best ring candidate is admitted as part of the ring network design. In doing this, all of the demand that is carried by that cycle is marked as having been loaded so that it will no longer be considered in subsequent iterations. If there is no further demand to be carried, or if no cycle was found to be able to carry any demand in this iteration, the design process terminates. Otherwise, the process iterates with the remaining network demand segments, considering all of the cycles again.

Macroscopically, both homogeneous and heterogeneous designs follow this process, but they differ in many important details, as described in subsequent sections of this chapter.Figure 29 shows the basic structure of RingBuilder™.

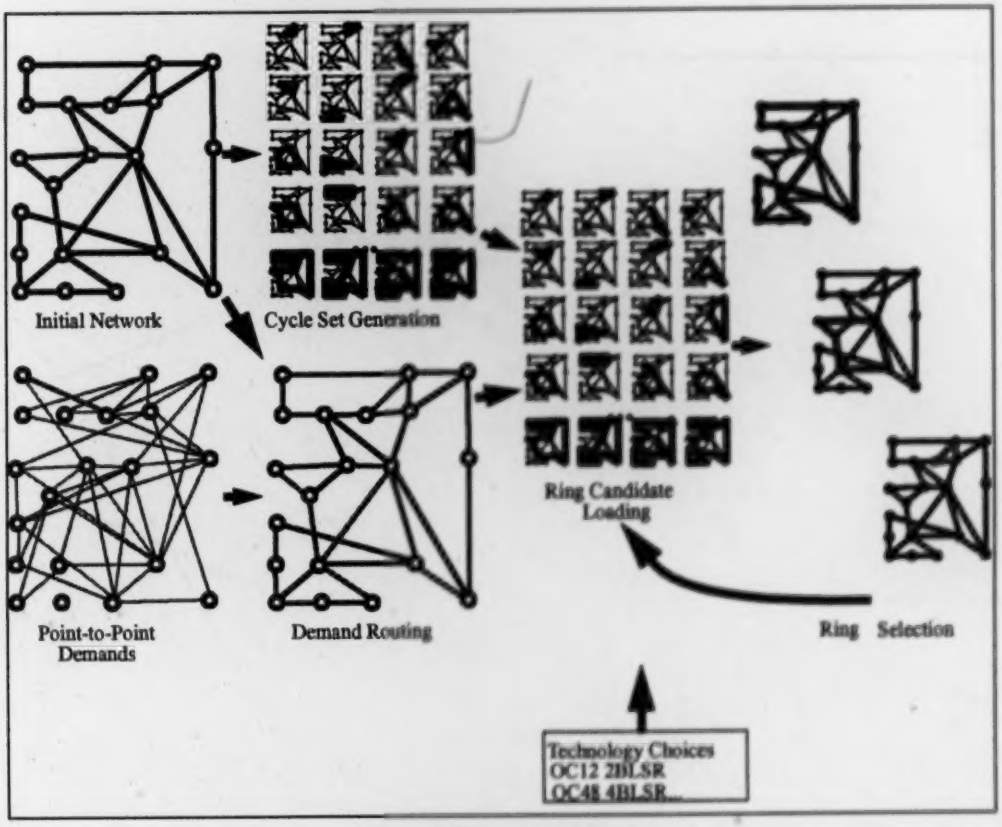


Figure 28: RingBuilder™

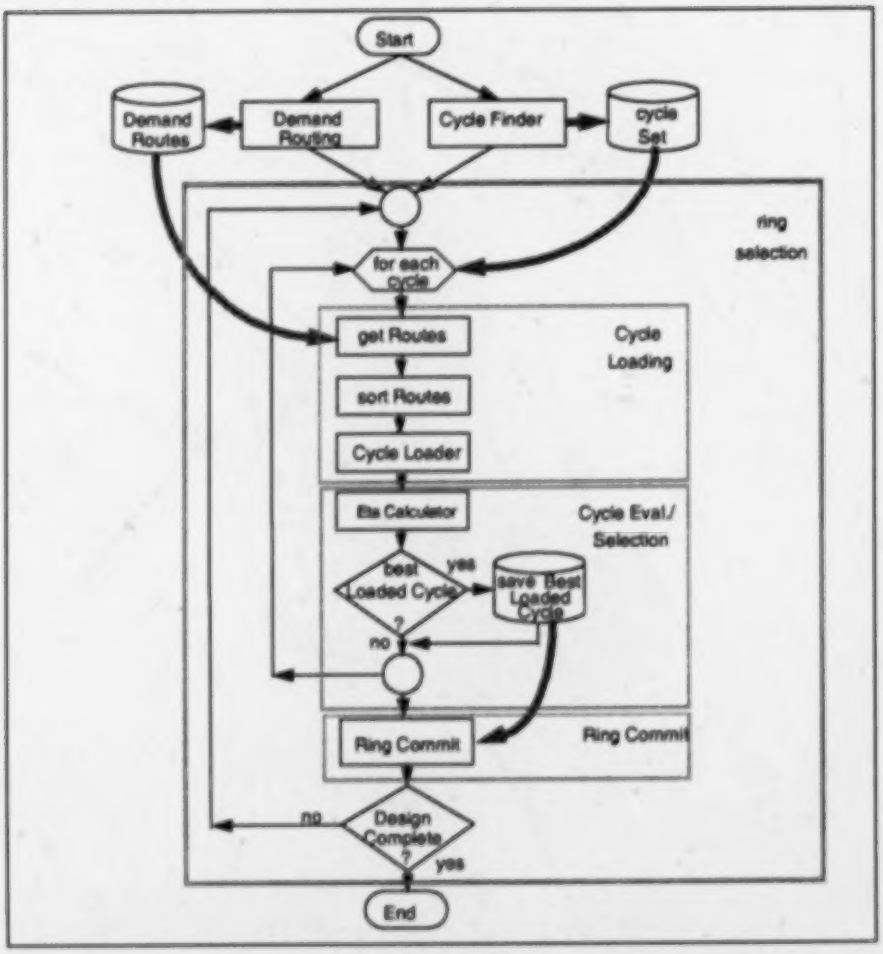


Figure 29: RingBuilder™ Iterative Ring Synthesis Framework

4.3 Cycle Finding

RingBuilder™ uses a depth-first search technique to determine all of the possible cycles on the graph representation of the network. The depth-first search is described in detail in [9]. Briefly, starting at an arbitrary initial node, the cycle finder sets out along an untried span emanating from the node and moving to a new node. This process repeats until a node that has already been visited is reached once more. The closed path is identified as a cycle candidate, and if unique, is stored as a member of the cycle set for the network. Otherwise, it is discarded. The cycle finder then moves back to the penultimate node

reached before the cycle candidate was identified, and tries to head out on another untried span, continuing until yet another node already visited is found. If there are no more untried spans at the penultimate node, the cycle finder retreats one node further and heads out on any untried spans at that node. This process terminates when the cycle finder has retreated to, and has tried all unvisited spans on, the initial node.

There are actually three distinct ways in which cycle finding can be used in RingBuilder™ (Figure 30). It can be used to find an entire cycle set, it can be used to find a partial cycle set (if the network is too extensive to discover the entire cycle set in reasonable time), or it can be used to incrementally discover cycles. This last option, first disclosed during initial RingBuilder™ development [9], has been used to investigate designs for extremely large networks. This method allows the RingBuilder™ framework to generate progressively better network designs as more and more cycles are discovered from which to choose in generating ring candidates.

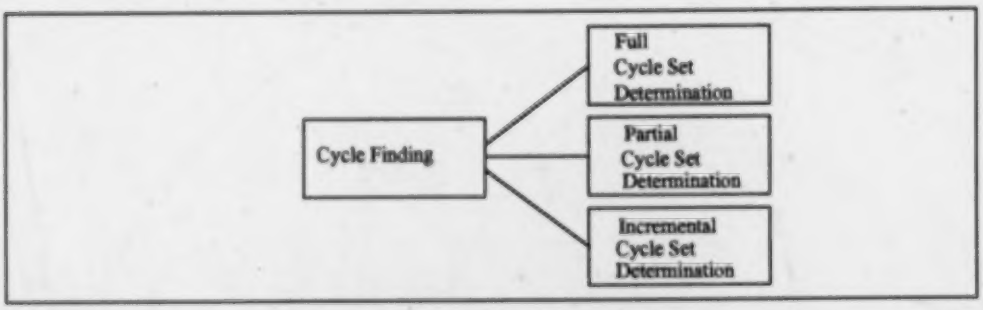


Figure 30: Cycle Finding Options

All of the experiments described in Chapter 5 use full cycle set determination.

4.4 Demand Routing

The RingBuilder™ framework can use shortest single physical path routing, shortest single logical path routing, shortest k-way physical path routing, and shortest physical k-way logical path routing (Figure 31). Experiments using all of these routing strategies are

described in Chapter 5.

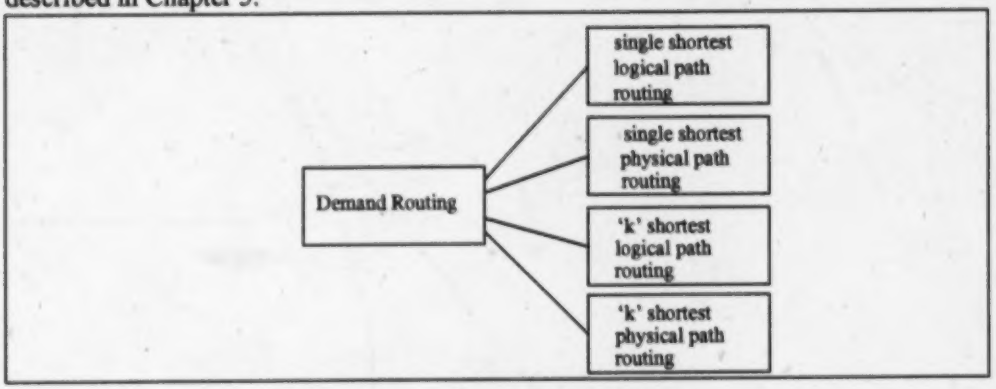

Demand Routing
single shortest logical path routing
single shortest physical path routing
'k' shortest logical path routing
'k' shortest physical path routing

Figure 31: Demand Routing Options

4.5 Ring Candidate Loading

The formation of ring candidates from cycles is critical to the generation of efficient designs. RingBuilder™ uses heuristic techniques to load the demand route segments in the formation of the ring candidates. The ring loading process is split into two separate functions: demand route sorting and demand route loading.

4.5.1 Demand Route Sorting Options

A set of demand routes to be loaded onto a cycle must be determined before that cycle is loaded. Generally, more demand routes are available for loading than that cycle can carry. As a result, they must be sorted in decreasing order of suitability for loading on the cycle in question. Two demand route sorting strategies are presented here: simple demand route sorting and complex demand route sorting.

RingBuilder™'s ***simple demand route sort heuristic*** orders, by decreasing cycle involvement, the demand routes that have at least one demand route segment in common with the cycle being loaded. *Cycle involvement* is defined as the bandwidth distance product of the not-yet-loaded demand route segments which intersect with the cycle's trajectory. The main premise of this heuristic is the hypothesis that the longer a large demand route is carried on a cycle, the better suited the demand route is for inclusion in the ring candidate formed from the cycle being loaded. The more of the 'well-suited' demand routes a ring candidate can carry, the better utilized and hence more efficient the resulting

ring candidate will be. This heuristic was developed as a simplification of the complex demand route sorting strategy presented next.

A ***complex demand route sorting heuristic*** was developed for RingBuilder™ in order to exploit the notions of cycle involvement, containment, and length of travel. The demand routes are first sorted in order of decreasing containment. *Containment* is defined as the ratio of the bandwidth-distance product of the not-yet-routed demand route segments which intersect with the cycle being considered to the total bandwidth-distance product of all of the not-yet-routed demand route segments. The demand routes that have the same containment figure of merit are sorted on the basis of decreasing cycle involvement. Those that have the same containment and the same cycle involvement are sorted on the basis of decreasing length measured in hops of travel around the cycle. This complex heuristic was the initial heuristic developed for the RingBuilder™ framework. It was formulated to prevent the depletion of ring candidate span capacity by a series of short, low bandwidth demands that would prevent the use of other spans of the ring and result in an inefficient ring candidate.

4.5.2 Demand Route Loading Options

The second phase of the ring candidate loading process in RingBuilder™ is the demand route loading stage. After the demand routes have been appropriately sorted, one of two RingBuilder™ demand route loading heuristics is invoked: capture-biased demand route loading or balance-biased demand route loading.

Capture-biased demand route loading attempts to find and load those not-yet-routed demand routes which can be fully transported from source to destination on the ring under consideration. Demand routes which do not qualify because they cannot be fully loaded on the ring candidate are set aside so that they may be subsequently loaded onto suitable ring candidates. The net effect of this heuristic is to generate loaded ring candidates that have the minimum number of inter-ring demand transitions, and hence the minimum terminal cost. Capture-biased demand loading is particularly suited to metropolitan ring designs where interface costs are high relative to transport costs.

Balance-biased demand route loading strives to load any and all demand route seg-

ments that are in common with the cycle. This heuristic attempts to maximize the fill of the ring candidate being implemented, and is particularly useful in the synthesis of long-haul transmission systems where transport cost is high compared to terminal costs.

4.5.3 Active Node/ Glass Through Node Optimization

RingBuilder™ solves the potentially very complex problem of deciding where active and inactive nodes are placed implicitly through the use of the heuristic techniques just described. The potential complexity results from the combinatorial explosion that would occur if a brute force technique was used, one that searched for proper active node determination by examining all possible ring candidates formed by the combinations of all possible active nodes. The demand route sorting heuristic, combined with the demand route loading heuristic, implicitly specify where the active nodes containing ADMs will be in the resulting ring candidate. After all demand routes have been examined for loading on the ring candidate, the active node locations are specified and the remaining nodes are by default deemed to be inactive nodes. The current version of RingBuilder™ does not distinguish between glass-through and pass-through nodes. All inactive nodes are assumed to be glass-through nodes since RingBuilder™ does not consider regenerator elements in the rings it forms.

4.6 Choosing Rings

The ultimate goal in the multiple SHR design problem is to find the most cost-effective network design that accommodates all of the demands. The RingBuilder™ framework supports two distinct optimization strategies for choosing ring candidates in the SHR design: balance vs. capture optimization and direct cost optimization via specific progress. The development of balance vs. capture optimization was initially based on the discovery of the fundamental balance and capture efficiency properties described in the previous chapter. It is an indirect cost optimization technique because it optimizes architectural properties that are indirectly related to the cost of the network being generated. Direct cost optimization via specific progress is a new optimization technique that is used in the RingBuilder™ framework. Direct cost optimization was initially developed as a result of a

desire of the research team at TRLabs to synthesize heterogeneous (multiple technology) SHR designs. As will be shown in the experimental results in Chapter 5, direct cost optimization via specific progress is equally well suited to single technology optimization and serves as a complementary technique to balance vs. capture optimization. Both optimization strategies are now described.

4.6.1 Balance vs. Capture Optimization

The two main cost drivers in the multiple ring design problem are the cost of transport of the carried bandwidth and the cost of loading and unloading the carried bandwidth from the ring. These two characteristics can be examined by measuring the capacity balance and traffic capture of the constituent rings of the design [9,10]. Balance and capture are opposing forces in the ring design problem. Optimizing for balance results in lower capture efficiency, whereas optimizing for capture compromises the balance efficiency. In the course of this work it was hypothesized that the minimum cost design can be found by using balance vs. capture optimization. This is done by searching for a design where the relative emphasis on balance and capture for each constituent ring is based on the relative cost of transport versus access. The efficiency expressions for balance and capture from Chapter 3 can be linearly combined to calculate a single figure of merit for a ring in the design, using a user-specified weighting factor, α, which is called the balance bias factor. The hybrid expression becomes:

$$\eta = \alpha \times balance + (1 - \alpha) \times capture \tag{12}$$

The planner, by specifying α, can generate a series of SHR designs to explore the solution space (each entire design representing a point in the solution space), ranging from pure balance to pure capture optimized designs and designs based on varying proportions of balance and capture weightings. The main disadvantage of the method is that it can take many runs to determine the appropriate value for the balance factor α. Even if the proper balance factor could be determined based on span and node costs, the planner would still want to generate designs around the appropriate value in order to minimize the chance of missing a local minimum, something that can occur because of the greediness of the optimization strategy. Balance vs. capture optimization is shown in Figure 32.

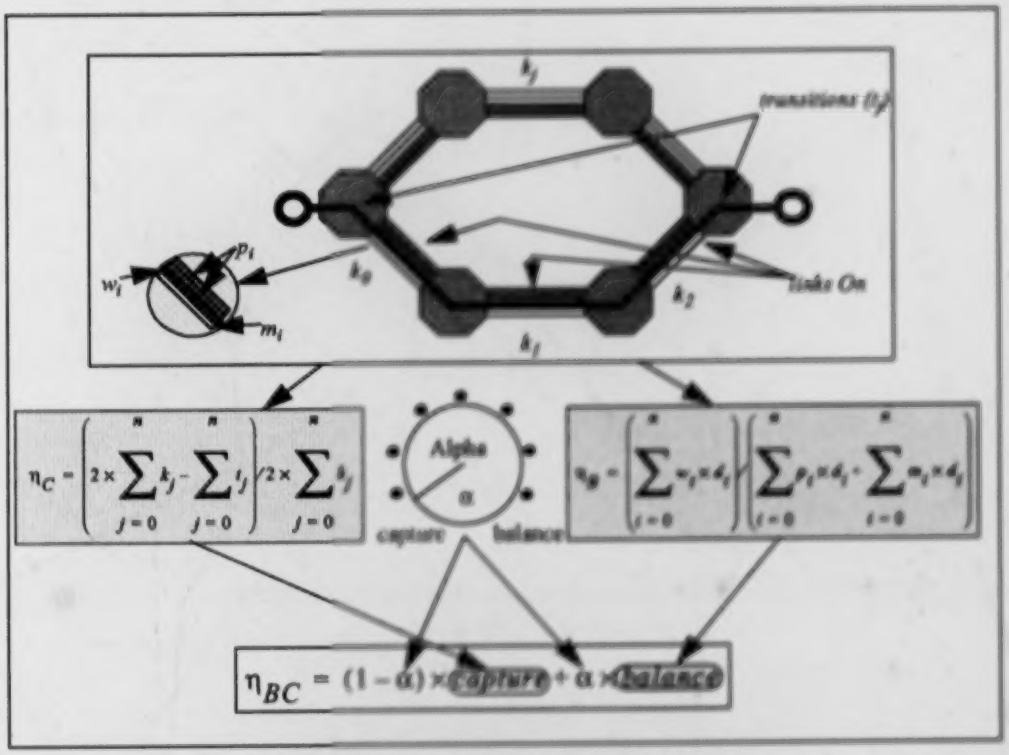


Figure 32: Balance vs. Capture Optimization

4.6.2 Direct Cost Optimization via Specific Progress

Direct cost optimization via specific progress directly evaluates the cost of implementation relative to the amount of progress towards the goal of a complete design. *Specific progress* is a measure of the amount of progress towards design completion the ring system under evaluation contributes relative to the cost of implementing that system. *Progress* in this context is a measurement of the amount of demand carried, based on the premise that the larger the amount of demand carried, the less that remains, and hence the fewer number of systems left to place. This is based on the fairly strong practical heuristic that if fewer systems are required to form a complete network design, then the network design will be more efficient and lower in cost, as each individual system in that design will be more highly utilized. Progress is measured in terms of the sum of the routed bandwidth-distance product of each of the demands carried by the system. The bandwidth-distance product is a measure of the transport system resource utilization of the demand as it

is carried from source to destination. The cost of implementing the system is the cost of the individual components required to construct the system. These components include the fibre in the spans, as well as the terminal equipment present in the nodes of the network. The terminal equipment includes common equipment such as power supplies and equipment bays, as well as provisionable equipment like add/drop port cards and opto-electronic modules. Specific-cost is a technology-independent way to compare ring systems. It has the advantage over balance vs. capture optimization in that rings of different technology types can be compared. Specific progress is summarized in Figure 33.

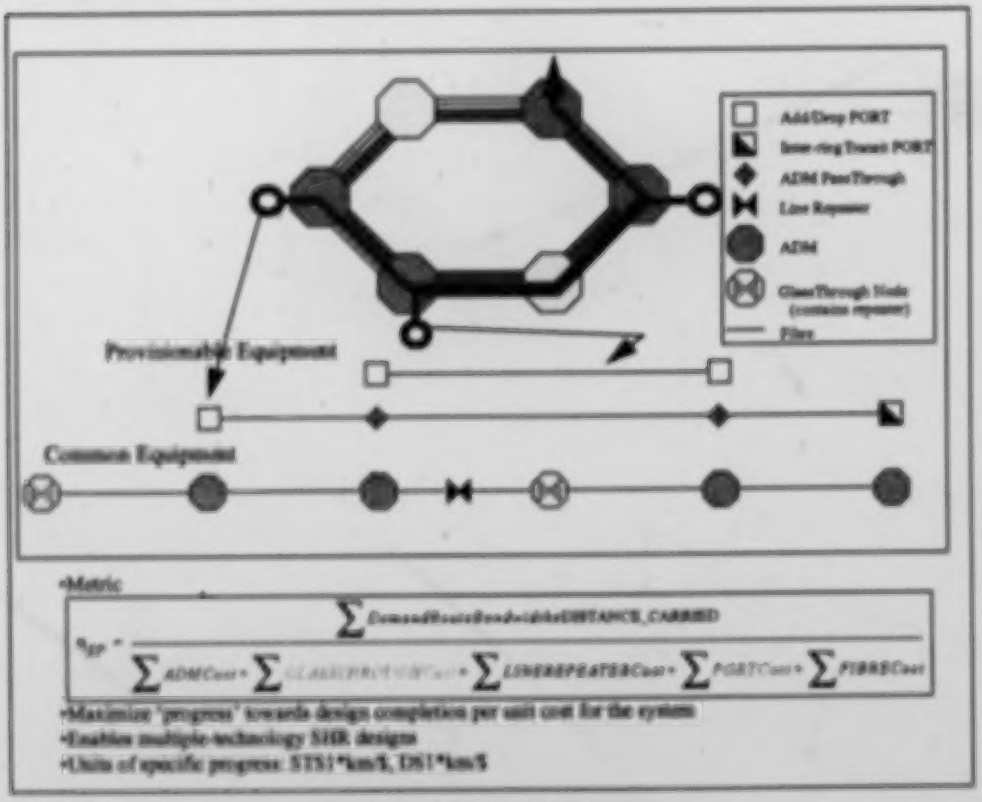


Figure 33: Specific Progress

4.7 RingBuilder™ Detailed Description

4.7.1 Single Technology Balance vs. Capture Optimization

Single technology (i.e., designs that are homogeneous in ring type, such as all 2-fibre BLSR OC-48) balance vs. capture optimization takes the basic greedy iterative framework and uses the concepts of balance and capture to evaluate loaded ring candidates. Rings are selected based on a hybrid of the individual balance and capture figures of merit calculated for the loaded cycles. The weighting factor α determines the relative bias of the balance figure of merit and the capture figure of merit, as described in Chapter 3. The single technology balance vs. capture optimization framework is shown in Figure 34. Note that the different demand bundling options as well as the different routing strategies are shown. These are all options the designer may manipulate when generating a network design.

Two route sorting options are available for the design synthesis, simple and complex. The simple route sorting strategy orders routes by decreasing progress. The complex route sorting strategy first sorts routes by decreasing ratio of links on the cycle relative to the total number of unrouted links for the route. In case of any ties, the tied routes are sorted by decreasing progress. In case of a tie in progress, the tied routes are sorted by decreasing unrouted hop count on the cycle under consideration.

The complex route sorting technique proves to be a superior heuristic (see Chapter 5), but is much more computationally involved. Because this route sorting is implemented for every cycle for each iteration, it can increase the real-time computational requirements.

Once route sorting is completed, the cycles are loaded using one of three possible cycle loaders (depending on the technology chosen for the design). It is not possible to directly compare the balance and capture figures of merit from loaded cycles that have been constructed using different equipment technologies. Balance vs. capture optimization is therefore an inherently single technology optimization framework. The three available loaders are UPSR, BLSR Balance, and BLSR Capture. Each of these loaders is fully configurable for all line rates, 2 or 4 optical fibres, and various cost parameters (common equipment cost, provisionable line card equipment, and fibre cost). Since regenerator costing is not currently implemented, regenerators have to be considered manually via a post-

processing step.

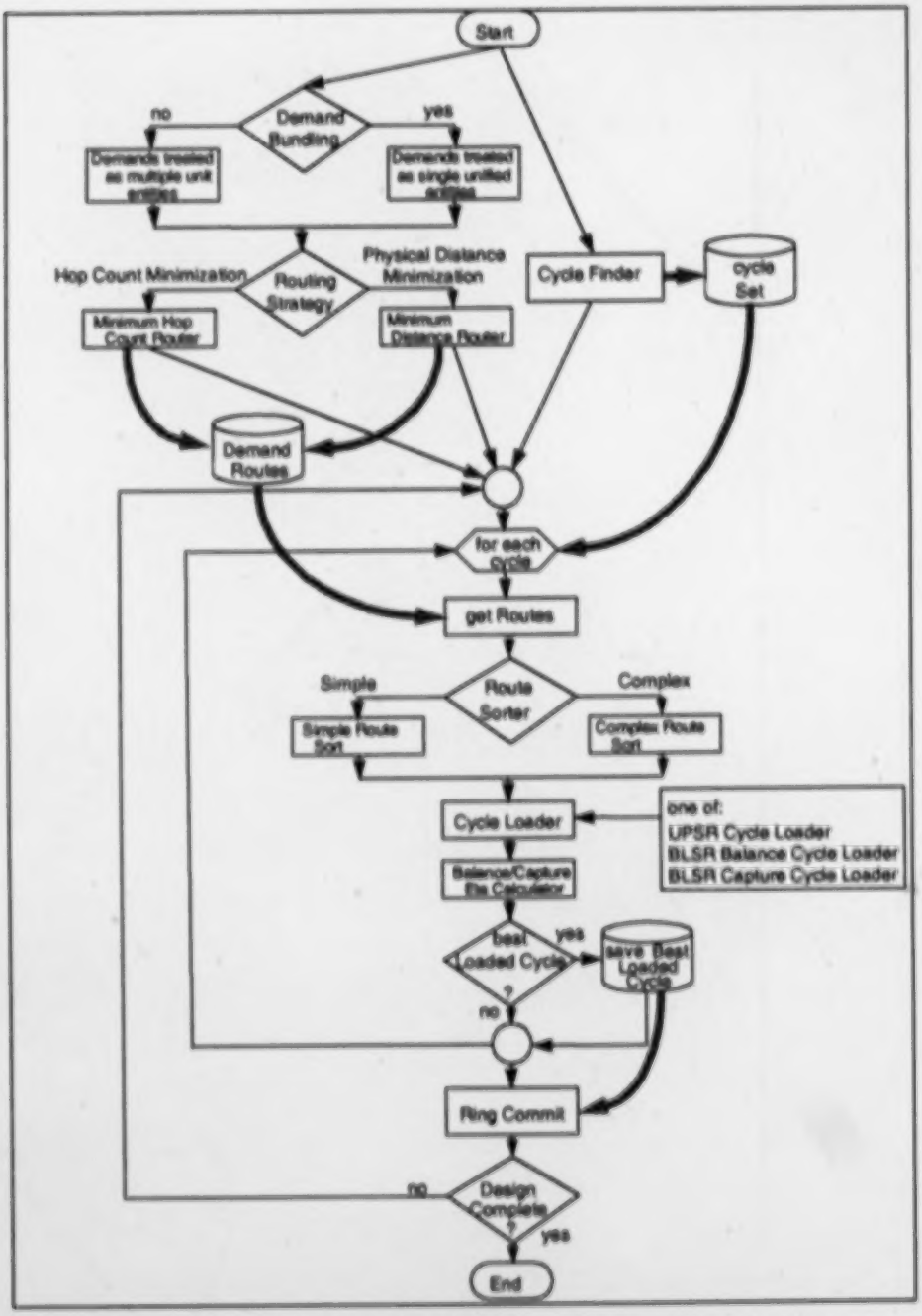


Figure 34: Single Technology Balance vs. Capture Optimization

The UPSR loader takes demand segments and assigns time slots for each demand all the way around the ring. Once a time slot is assigned for a demand, it is not available anywhere else on the ring. This is inefficient relative to BLSR span reuse capability. It is an advantage in situations where broadcasting of the payload data to more than one node is required, such as in some drop and continue node protection scenarios or multimedia distribution.

The BLSR balance loader takes demand segments and assigns them time slots on the spans that need to carry the demand. The balance functionality attempts to load any segment of a demand, even if all segments of a demand coincident with the cycle are unable to be loaded. This loader maximizes the utilization of the transport capacity at the expense of potential excess inter-ring transiting as capacity-blocked spans are bypassed by transporting the demand on subsequent ring system choices.

In contrast, the BLSR capture ring loader only accepts a demand for loading onto a system if all unrouted links of that demand that are common to the system can be loaded by the system. Otherwise, the demand is rejected and left for another subsequent system to load. The BLSR capture ring loader minimizes the amount of inter-ring transiting by enforcing this all-or-nothing policy. It does so at the expense of potential overall system fill since demand segments that could feasibly be loaded may be rejected if they are related to other segments that have been rejected due to system or network capacity constraints.

In each iteration, each cycle candidate is loaded and the one with the best balance vs. capture figure of merit is nominated to be a ring in the system. The ring commit process captures the loaded cycle, now called a ring. It records the path, ADM placement, provisionable element counts, fibre utilization, and a detailed description of each of the demand routes fully or partially loaded onto that system. The ring commit process also removes the carried demand segments from further consideration by the design system. If no more uncarried demand is left in the system, or if any further demand cannot be carried by rings in the network, the design process terminates.

4.7.2 Multiple Technology/Direct Cost Optimization

The multiple-technology/direct cost optimization framework builds on both the generic iterative framework as well as the single technology framework, adding function-

ality to allow the evaluation of multiple simultaneous technology choices in the cycle loading process. The major benefit of this technique is that it provides an automated, integrated method for the efficient design of a ring-based network, using real-world constraints including system capacities and actual system component costs. The framework is still a greedy one and thus produces near-optimal rather than optimal designs. Because the actual costs of the components involved are considered in the evaluation process, the best possible system choice is made from the perspective of cost per unit progress towards the final design.

The basic framework is outlined in Figure 35. It is very similar to the balance vs. capture optimization framework with the exception of the multiple parallel cycle loaders and the substitution of specific progress for the balance vs. capture calculation. Another inner loop is added to the iterative framework so that each cycle is evaluated after it is loaded by each technology option. Any number of several technology options may be considered.

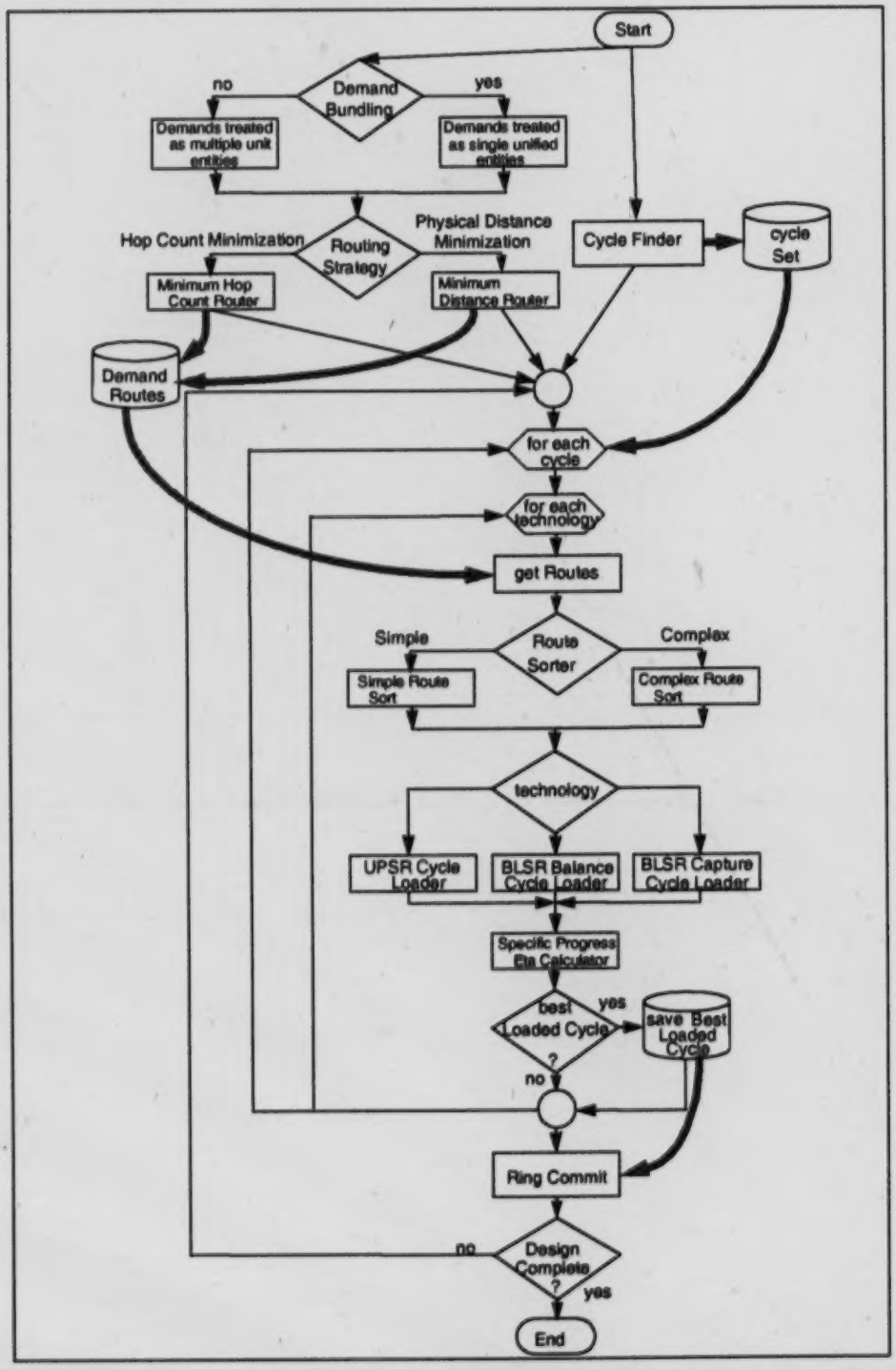


Figure 35: Specific Progress Direct Cost Optimization Greedy Iterative Framework

4.8 Design Example

A 2BLSR12 single technology example design will be synthesized for the network and corresponding point-to-point demand set shown in Figure 36. All span lengths are unity. The system demand management unit is STS1 and thus the demands are expressed in STS1 units. Modularized distance-weighted balance optimization is used. From(9) in Chapter 3:

$$\textit{modularized distance-weighted balance} = \left(\sum_{i=0}^{n-1} W_i \cdot d_i\right) \Big/ \left(\sum_{i=0}^{n-1} P_i \cdot d_i + \sum_{i=0}^{n-1} M_i \cdot d_i\right) \quad (13)$$

A 2BLSR12 with STS1 demand management has 6 STS1s of capacity on each of its spans.

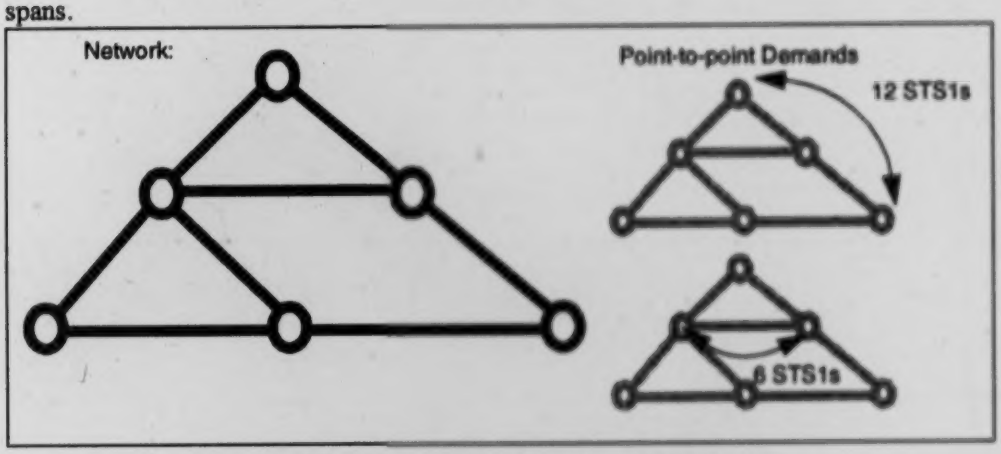


Figure 36: Example -- Network and OD Pairs

First, the demands are shortest path routed, as shown in Figure 37.

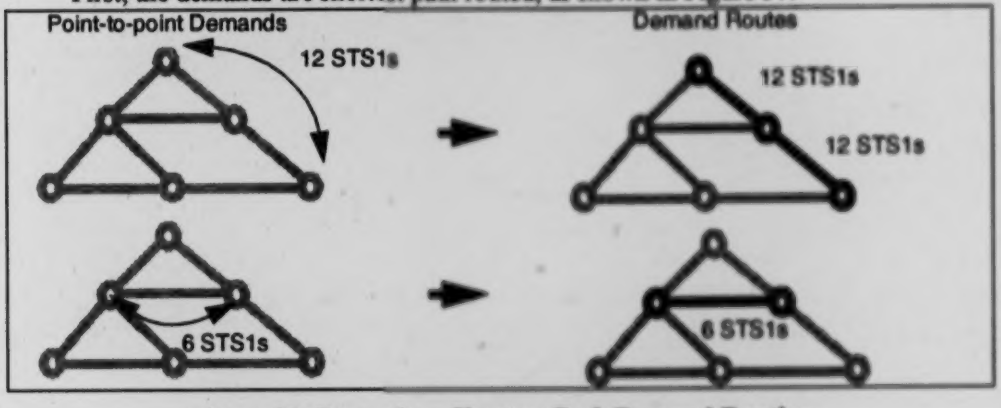


Figure 37: Example – Shortest Path Demand Routing

Then the cycle finder is invoked and the cycle set is found as shown in Figure 38.

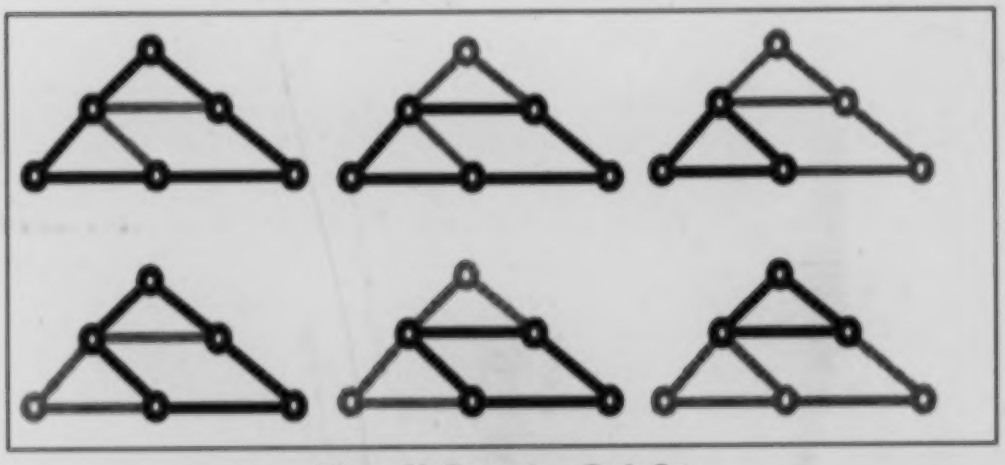


Figure 38: Example -- Cycle Set

Now the iterative ring selection process starts. For the first iteration, the cycles are loaded as shown in Figure 39. After cycle loading, each of the ring candidates is evaluated, and the best is selected, as shown.

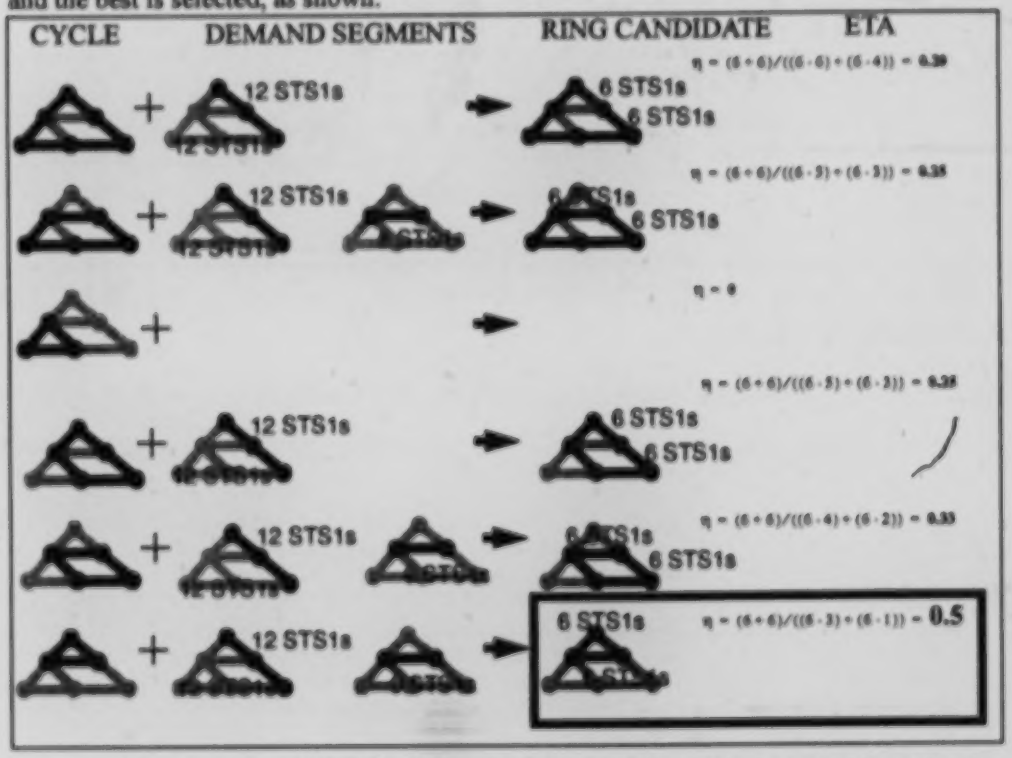


Figure 39: Example -- Ring Candidate Loading and Ring Selection for Iteration 1

The demand segments carried by the ring are removed from further consideration, and the cycles are loaded with the remaining demand segments forming new ring candidates as shown in Figure 40, and the next ring is selected. The third and last iteration is shown in Figure 41. The final ring set is shown in Figure 42.

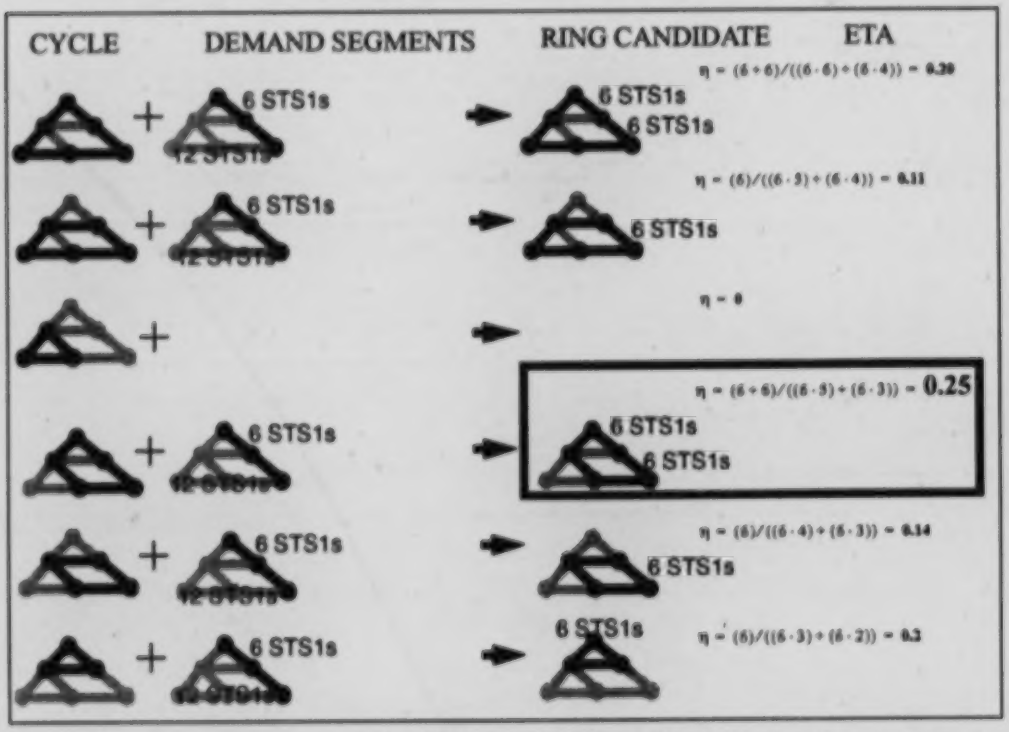


Figure 40: Example -- Ring Candidate Loading and Ring Selection for Iteration 2

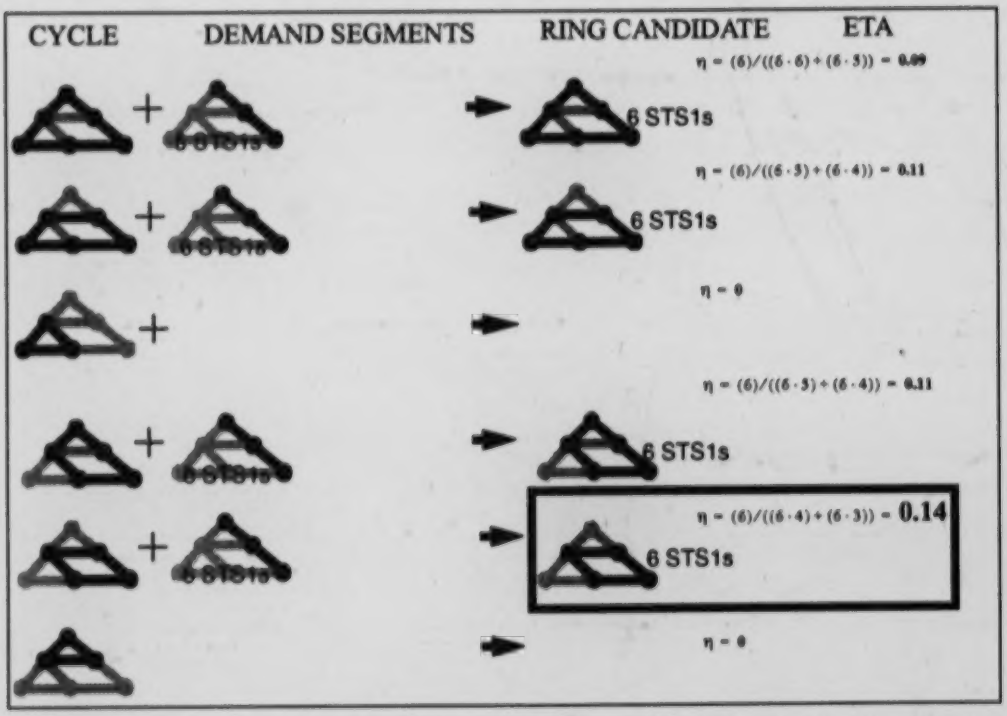


Figure 41: Example -- Ring Candidate Loading and Ring Selection for Iteration 3

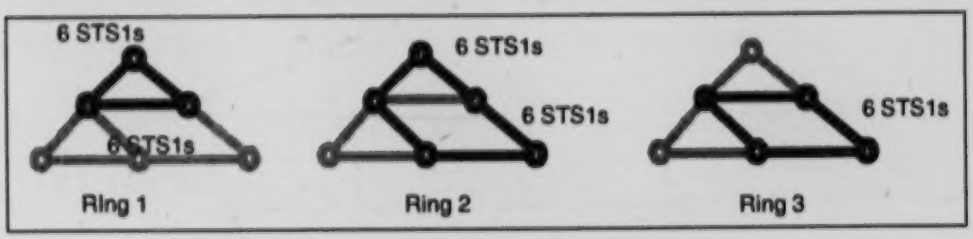


Figure 42: Example -- Rings in Final Design

A RingBuilder design example using the direct cost optimization via specific progress technique is described in Appendix B.

4.9 Conclusion

This chapter introduced RingBuilder™, a new synthesis technique, from a functional and conceptual level. The two new optimization strategies developed in the course of the RingBuilder™ project, balance vs. capture optimization and direct cost optimization via

specific progress were described. RingBuilder™'s method of addressing the design considerations is highlighted in Figure 43. Further technical details of the data structure organization and functional module implementation of RingBuilder™ can be found in [19].

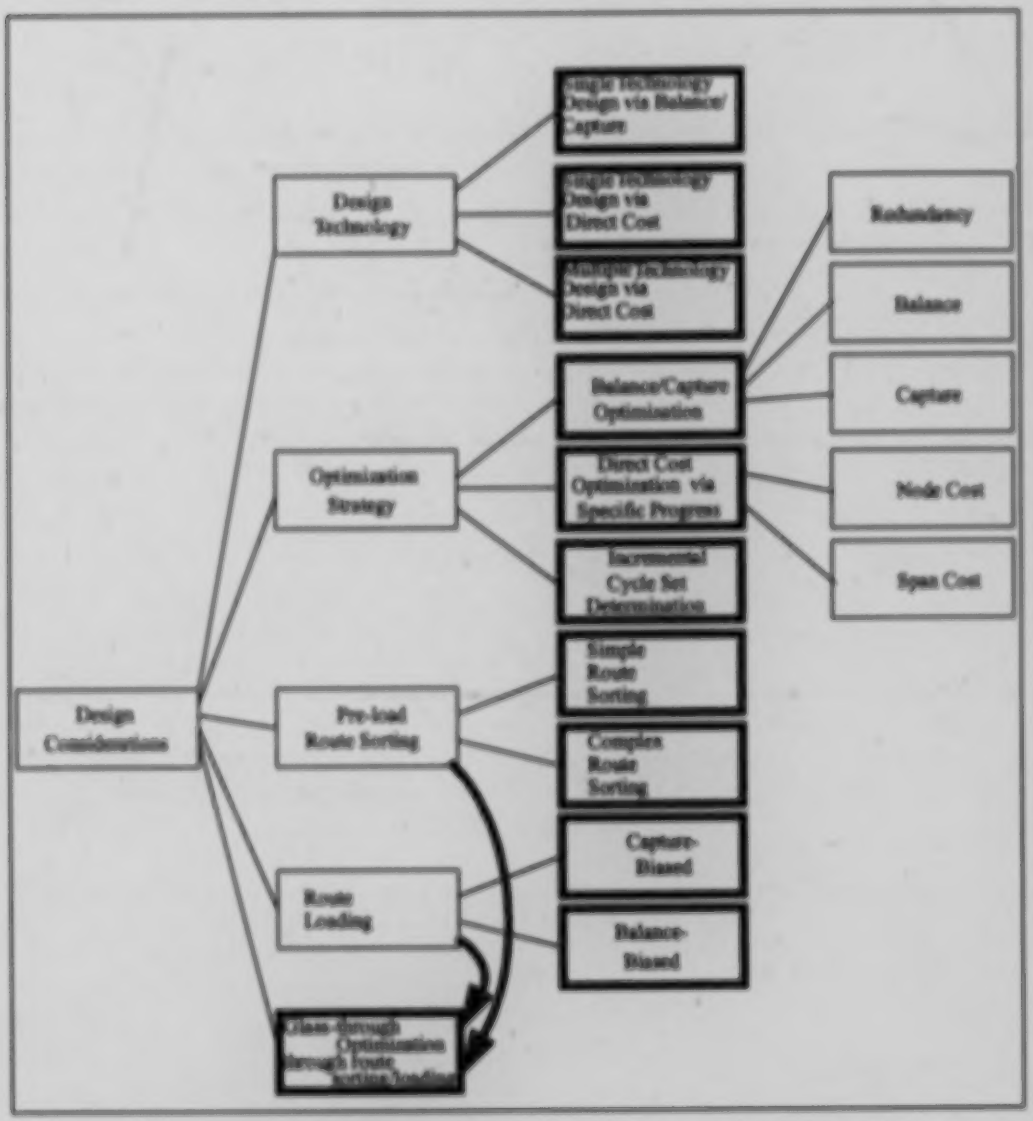


Figure 43: Design Considerations as Addressed by RingBuilder™

5. Experimental Results

This chapter documents the performance of RingBuilder™ with several 'real-world' networks. The most effective combinations of options for each of the network types studied are discovered, certain characteristic properties of the networks in the context of ring network design are identified, and parameter optimization recommendations, based on network type, are presented.

Two types of networks are used in this chapter, metropolitan and long-haul. Most transport networks can be classified as either metropolitan or long-haul based on their span lengths. Metropolitan networks have relatively short spans, thus the majority of the cost in a ring network design is in the terminal equipment at the nodes. Long-haul networks have long spans so that a significant part, if not the majority, of the network cost is in line-related fibre and repeater costs. Metropolitan and long-haul networks represent characteristic network boundary conditions. Most real-world networks have attributes that lie between these two extremes.

5.1 Test Networks

Three different networks are used in the experiments described in this chapter: two metropolitan networks (netJ and netY) and one long-haul network (netM). NetJ is shown in Figure 44. To demonstrate the use of the RingBuilder™ framework for studying the effects of differing network topologies, four variations of netY were examined: minimal connectivity (netYm), sparse connectivity (netYa), nearest neighbor connectivity (netY) and full connectivity (netYb). The netY networks are illustrated in Figures 45 to 48. NetM is shown in Figure 49.

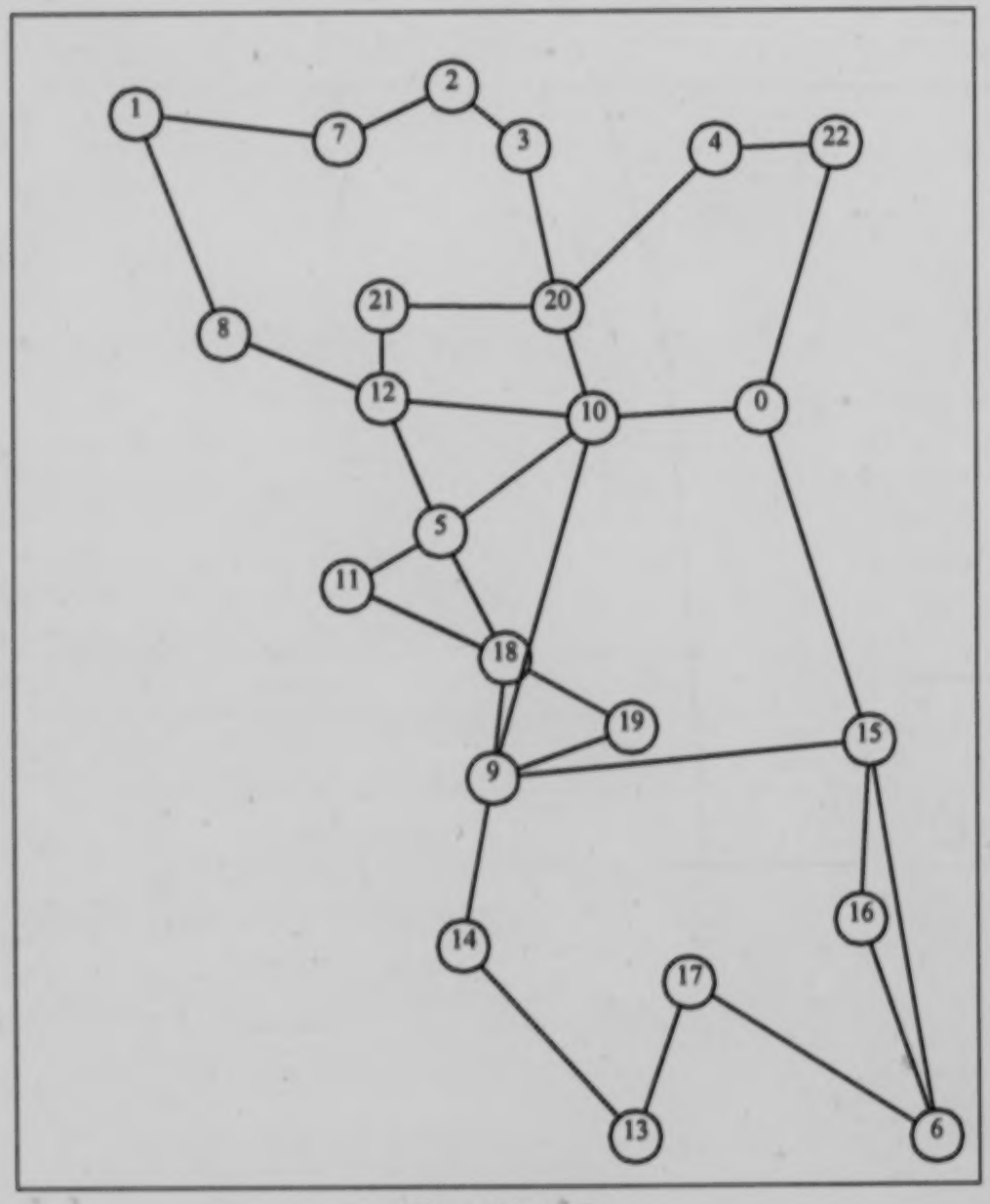


Figure 44: netJ

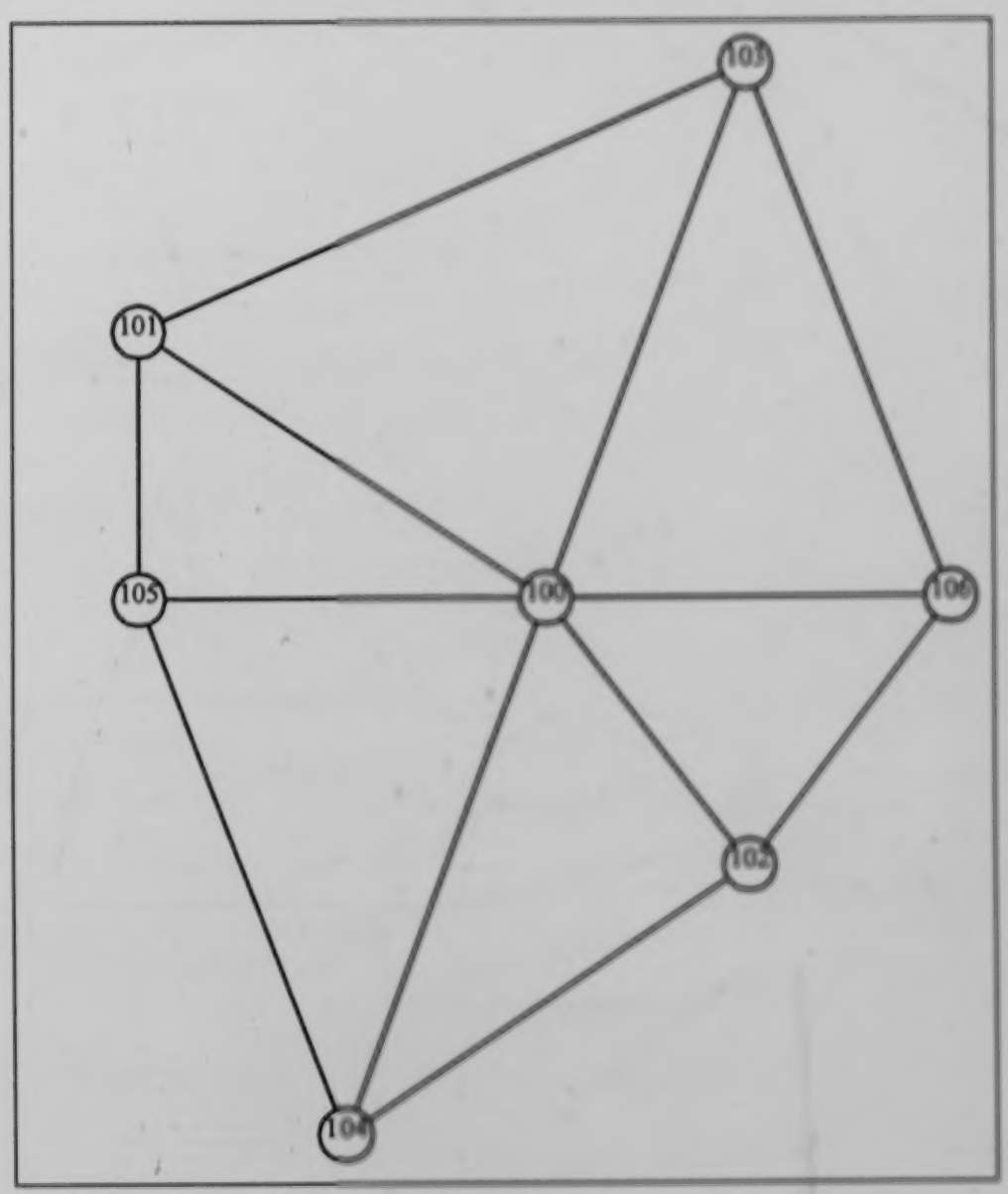


Figure 45: netV

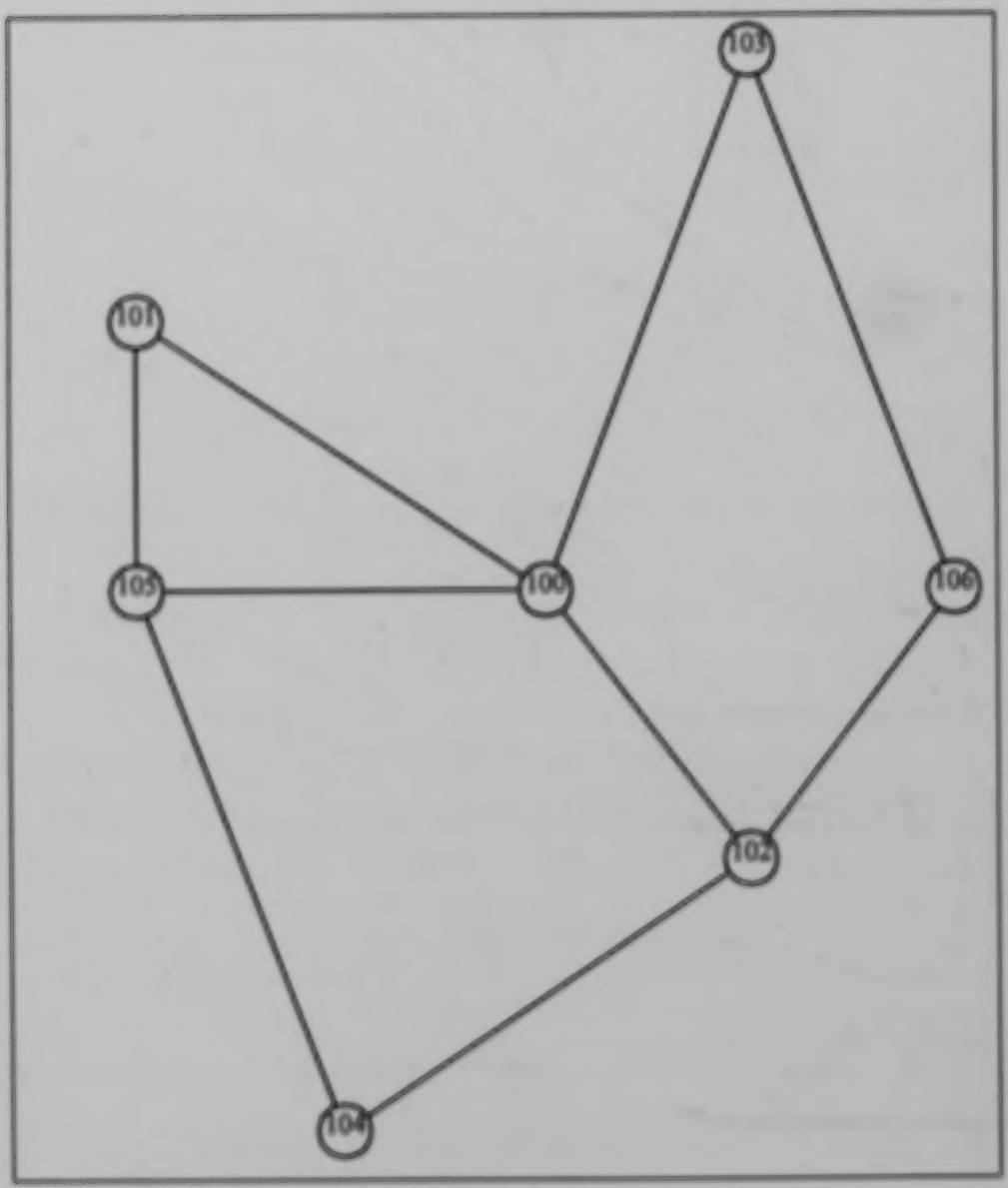


Figure 46: netYa

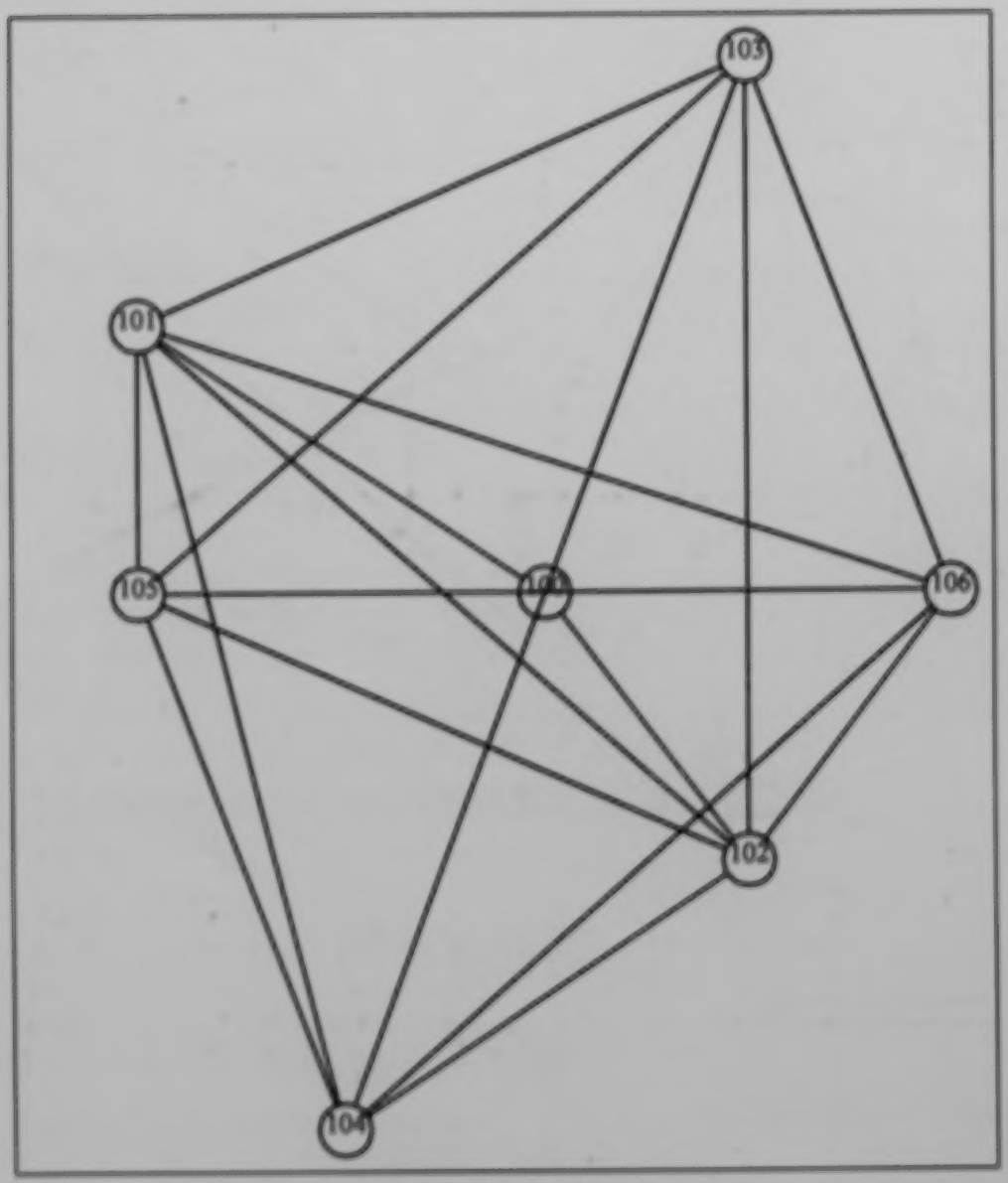


Figure 47: netYb

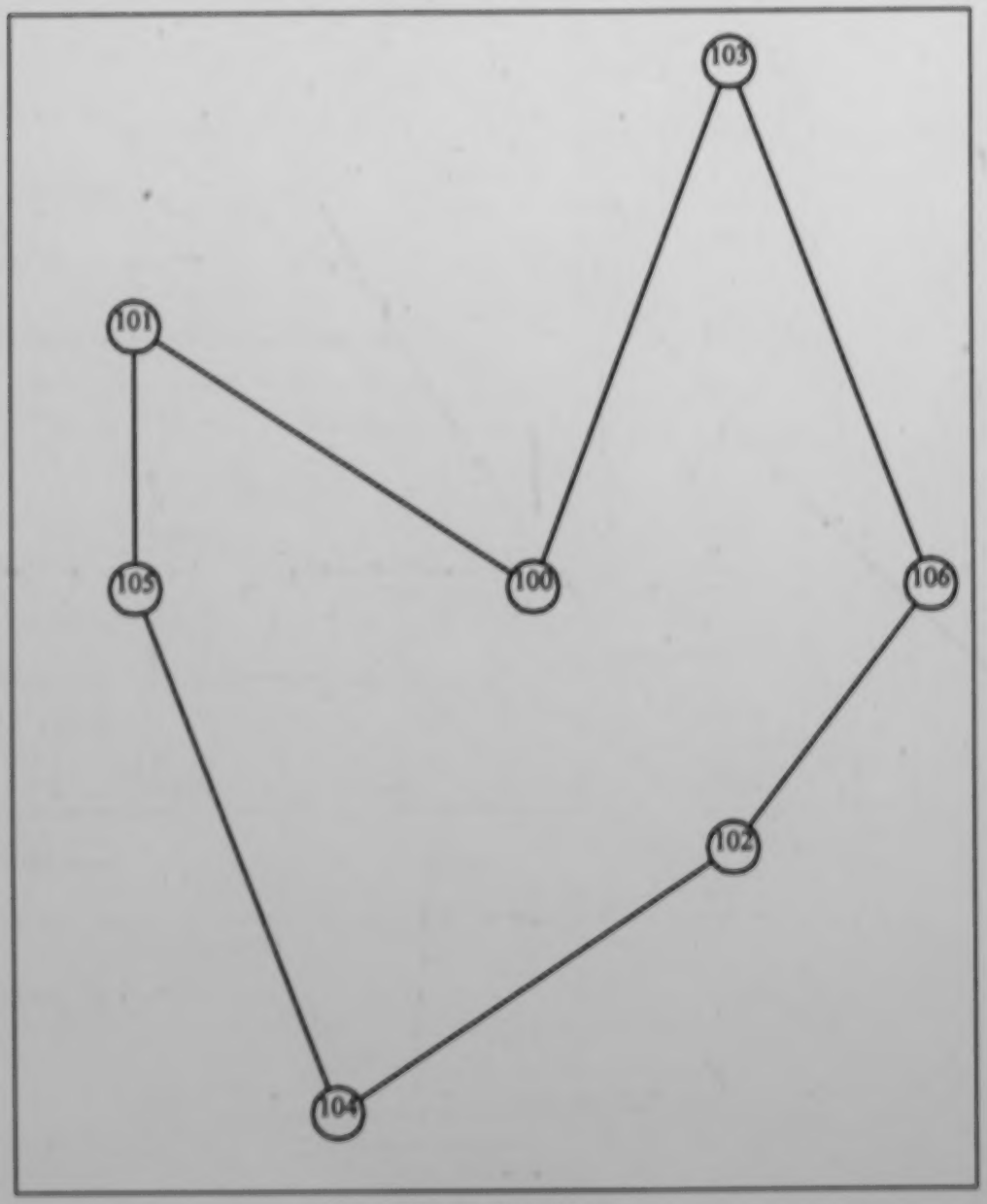


Figure 48: netYm

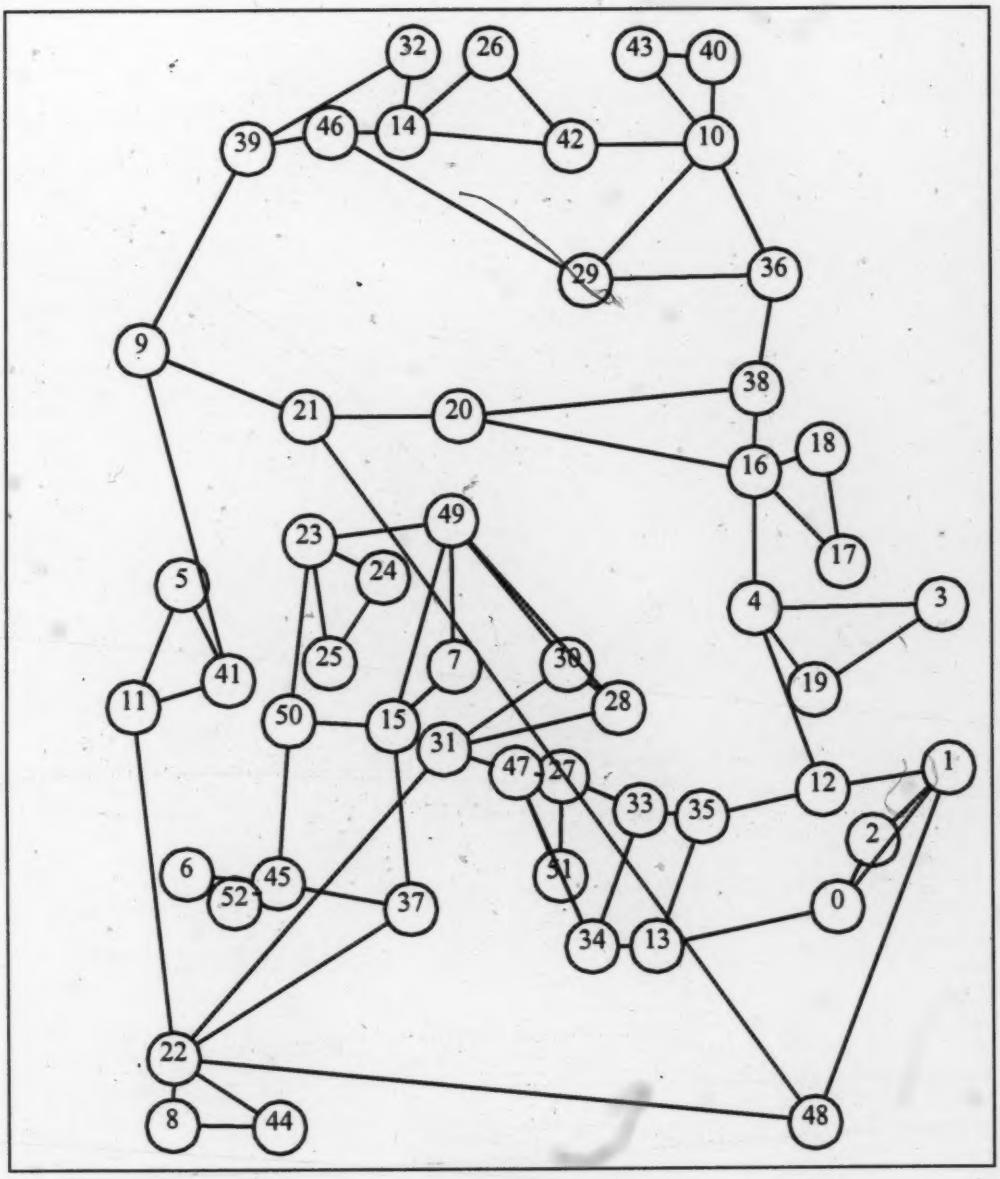


Figure 49: netM

Characteristics of the networks used are summarized in Table 3.

Table 3: Experimental Networks

Network	nodes	spans	span distance (km)	cycles	total demand
netJ	23	32	236.5	178	1547 DS1s
netM	53	79	15692.0	58893	858 STS1s
netY	7	12	24.4	31	1456 DS1s
netYa	7	9	17.0	6	1456 DS1s
netYb	7	21	57.1	1172	1456 DS1s
netYm	7	7	13.6	1	1456 DS1s

5.2 Cost Models

The cost models for each ring technology are shown in Table 4. The system attributes can be divided into three categories: system type, system capacity, and system cost. Both BLSRs and UPSRs are modeled. System capacities include the optical line rate, maximum ADM count, and maximum ADM add/drop count. System costs are the cost of the common ADM equipment, the cost of the provisionable add/drop and transit equipment, and the per unit distance cost of the fibre, including the span repeater equipment.

In addition to UPSR and BLSR systems, 1+1 diversely routed systems are modeled through the use of UPSR ring systems with only two ADMs.

Table 4: Technology Cost Models

System Technology	System Capacity			System Cost Components			
	Demand Management Unit	Max # ADMs	Max ADM Add/ Drops	ADM Common cost ($)	ADM Add/ Drop Cost ($)	ADM Transit Cost ($)	Fibre Cost ($)/ km
2BLSR12	DS1	16	336	70000	70/DS1	900/DS1	100
2BLSR48	DS1	16	1344	140000	70/DS1	900/DS1	100
2UPSR12	DS1	16	336	50000	70/DS1	900/DS1	100
4BLSR12	DS1	16	672	100000	70/DS1	900/DS1	100
4BLSR48	STS1	16	96	160000	2000/STS1	5000/STS1	3000
2BLSR192	STS1	16	192	320000	2000/STS1	5000/STS1	3000
2UPSR192	STS1	16	192	240000	2000/STS1	5000/STS1	3000
2UPSR48	STS1	16	48	160000	2000/STS1	5000/STS1	3000
2UPSR48	STS1	2	48	100000	2000/STS1	5000/STS1	3000
2UPSR192	STS1	2	192	220000	2000/STS1	5000/STS1	3000
2BLSR12	DS1	2	336	70000	70/DS1	900/DS1	100

5.3 Experimental Test Cases

The experimental cases are shown in Table 5. Test case23 and test case26 are the baseline cases for netJ and netM, against which the results are compared for every test suite except suite 10, which investigates the effect of network topology modification on SHR network design. The designs for these two test cases were generated using the following default options: demands treated as bundled entities, single shortest path routing, hop count minimization, direct cost optimization, simple pre-load route sorting, and capture-biased route loader. This set of options generates reasonable designs (as they are the options an experienced planner would use in generating a design manually). The rings produced for the case23 and case26 designs are shown in Appendix C. 2BLSR12 with DS1-level demand management is used for netJ, 4BLSR48 with STS1-level demand management is used for netM, and 2BLSR12 with DS1-level demand management is used for

netY/Ya/Yb/Ym. The experimental regimen perturbs only one variable at a time, so that its effect on the resultant design can be clearly seen.

The dependence of the design on the network topology is examined through fully-connected, intermediately-connected, and minimally-connected test networks. Network node and span constraints, the effects of preventing inter-ring transiting at a node, and different working path placement considerations are studied. Two strategies for demand bundling are investigated. Demand routing strategy is explored, looking at two ways of distributing equal length demand routes and two ways of optimizing demand routes. System technology considerations are examined by studying two types of ADM constraints, including ADM port count and active ADM count limits. Design considerations explored with the RingBuilder™ framework include design technology, design optimization strategy, pre-load route sorting strategy, and cycle route loading strategy. These test cases exercise all of the main features of RingBuilder™.

Table 5: Summary of Experimental Test Cases

Test Suite	Network	Test Case	Network Considerations					Working Path Placement Considerations					System Technology Considerations		Design Considerations								NOTES
			Network Topology Constraint					Demand Bundling		Demand Routing Strategy			ADM Constraints		Design Technology		Optimization Strategy		Pre-Load Route Sorting		Route Loading		
										Demand Route Distribution		Demand Route Optimization											
			Minimally Connected	Sparsely connected	Nearest neighbor connected network	Fully Mesh Connected	Node Transition Constraint	Demands treated as unified entities BUNDLED	Demands treated as multiple in-dependent units UNBUNDLED	k-way split	single shortest path	hop count minimization	physical distance minimization	ADM Port Count	Active ADM Count	Single Technology	Multiple Technology	Balance vs. Capture	Direct Cost	Simple	Complex	Capture-Based	Balance-Biased	
1	netJ	25		✓				✓			✓	✓				✓			✓	✓		✓		BASELINE CASE
	netJ	24		✓					✓		✓	✓				✓			✓	✓		✓		
	netJ	25		✓					✓	✓		✓				✓			✓	✓		✓		
	netM	26		✓				✓			✓	✓				✓			✓	✓		✓		BASELINE CASE
	netM	27		✓					✓		✓	✓				✓			✓	✓		✓		
	netM	28		✓					✓	✓		✓				✓			✓	✓		✓		
2	netJ	29		✓				✓			✓		✓			✓			✓	✓		✓		
	netM	30		✓				✓			✓		✓			✓			✓	✓		✓		
3	netJ	31		✓				✓			✓	✓				✓		✓		✓		✓		
	netM	32		✓				✓			✓	✓				✓		✓		✓		✓		
4	netJ	33		✓				✓			✓	✓				✓			✓		✓	✓		
	netM	34		✓				✓			✓	✓				✓			✓		✓	✓		
5	netJ	35		✓			✓	✓			✓	✓				✓			✓	✓		✓		
	netM	36		✓			✓	✓			✓	✓				✓			✓	✓		✓		
6	netJ	37		✓				✓			✓	✓		✓		✓			✓	✓		✓		
	netJ	38		✓				✓			✓	✓			✓	✓			✓	✓		✓		
	netM	39		✓				✓			✓	✓		✓		✓			✓	✓		✓		
	netM	40		✓				✓			✓	✓			✓	✓			✓	✓		✓		
	netM	40a		✓				✓			✓	✓			✓	✓			✓	✓			✓	
7	netJ	41		✓				✓			✓	✓				✓			✓	✓			✓	
	netM	42		✓				✓			✓	✓				✓			✓	✓			✓	
8	netJ	43		✓				✓			✓	✓					✓		✓	✓		✓		
	netM	44		✓				✓			✓	✓					✓		✓	✓		✓		
9	netJ	45		✓				✓			✓	✓					✓		✓		✓		✓	
	netJ	45a		✓					✓		✓	✓					✓		✓		✓		✓	
	netJ	45b		✓					✓		✓	✓					✓		✓		✓	✓		
	netM	46		✓				✓			✓		✓				✓		✓		✓	✓		
	netM	46a		✓					✓	✓		✓	✓				✓		✓	✓			✓	
	netM	46		✓				✓			✓		✓				✓		✓		✓		✓	
10	netY	47			✓			✓			✓	✓				✓			✓	✓		✓		
	netYm	47m	✓					✓			✓	✓				✓			✓	✓		✓		
	netYa	48		✓				✓			✓	✓				✓			✓	✓		✓		
	netYb	49				✓		✓			✓	✓				✓			✓	✓		✓		
11	netY	50		✓				✓			✓	✓			✓	✓			✓	✓		✓		
	netM	51		✓				✓			✓	✓			✓	✓			✓	✓		✓		
12	netJ	52		✓				✓			✓	✓				✓			✓		✓		✓	
	netM	53		✓				✓			✓	✓				✓			✓		✓		✓	

5.3.1 Design Evaluation Criteria

Each of the networks generated by the test cases is examined with respect to the following evaluation criteria:

- total system cost
- span cost
- node cost
- ring count
- average hop count per ring
- average circumference per ring
- ADM count
- average number of ADMs per ring
- inter-ring transition count
- installed working bandwidth-distance product
- installed margin bandwidth-distance product
- installed protection bandwidth-distance product
- average specific progress
- average balance
- average capture
- average balance-capture at specified balance bias factor (if applicable)

5.3.2 Results and Discussion

The 67 test cases used to examine the various aspects of RingBuilder™ are grouped into 12 *test suites*. Each test suite examines the effect of a different option, comparing it to its baseline test case. Each of these options represents a distinct design issue for the planner.

Every test suite except for test suite 10, which investigates the modification of network topology on SHR design, uses case23 and case26 as baseline designs for netJ and netM respectively.

Test Suite 1:Demand Bundling/Demand Route Distribution

Test suite 1 compares the performance of RingBuilder™ with bundled demands vs. unit demands. It further compares the performance of k-way splitting of unit demands vs. single shortest path routing of unit demands. The test cases are summarized in Table 6. Note that it is not possible to k-way split bundled demands as the demand bundle is then

indivisible.

Table 6: Test Suite 1

Demand Bundling	Demand Routing	
	K-Way Split	Single Shortest Path
Bundled Demands	[illegible]	case23 case26
Unit Demands	case25 case28	case24 case27

Case24 demonstrates that using unit demands for the metropolitan network netJ decreases the overall system cost from $5.18M in case23 (the baseline case) to $4.71M in case24. The number of rings in the design drops from 11 to 10, and a corresponding reduction in the number of inter-ring transitions, from 1674 to 1558, occurs. The number of required ADMs drops from 48 to 43, and the margin capacity drops from 55000 to 46000 DS1-km. This shows that the average utilization of the rings increases relative to the baseline case and the resulting design is more efficient.

When k-way routing of the point-to-point demands is allowed in case25, the design cost becomes $4.85M, which is lower than baseline case23, but more expensive than the unit demand, single shortest path design. This can be attributed to an increase in the total number of rings in the design, and the attendant increase in the number of inter-ring transitions. This is somewhat surprising, as the use of k-way routing serves to distribute the demand more evenly across the network, and a more evenly distributed demand should result in a design with lower total installed bandwidth. Out of 147 OD pairs, 46 are impacted by splitting, affecting 271 out of 1547 unit demands. Of these, thirty-five 2-way splits affecting 271 units of demand and eleven 3-way split affecting 71 units of demand occur. Spreading out the demand route distribution in this case caused a peak span working capacity increase, which triggered an additional system to be deployed.

In case27 where the long-haul netM is considered, the application of unit rather than bundled demands results in a design cost of $371M compared to the case26 baseline of $411M. In the unit demand case, the number of required rings drops from 27 to 26, and the

number of inter-ring transitions drops from 1282 to 1198. When k-way demand routing is enabled in conjunction with unit demand processing in case28, the cost further drops to $370M in a 26 ring design. The demand route splitting process impacted 34 of 347 OD pairs affecting 98 out of 858 unit demands. There were thirty 2-way splits affecting 82 units of demand, three 3-way splits affecting 12 units of demand and one 4-way split affecting 4 units of demand. Note that in case28, the number of inter-ring transitions is slightly higher than in case27, increasing to 1202 from 1198. The number of ADMs drops to 105. The total installed working bandwidth-distance product increases to 786359 STS1-km from 779705 STS1-km. The installed margin actually drops from 751543 STS1-km to 589945 STS1-km, resulting in a total installed bandwidth of 2753608 STS1-km.

The minimum cost design has the highest average specific progress in both the netJ and netM contexts. The lowest cost design also has the maximum average balance. The lowest cost netM design does not have the maximum average capture. These results are as expected given the dominance of node costs in the metropolitan context and the dominance of line costs in the long-haul context.

Unit demand processing proves to be more efficient than bundled demand processing in the generation of cost-effective designs.

There does not seem to be a large advantage to the use of k-way demand routing relative to single shortest path routing in the two networks described here. In netJ, the design cost actually rises; in netM, the cost decreases, but only by 0.14%. The reason for this may be that k-way routing does not assist in generating better balanced and thus more efficiently utilized rings. In the long-haul context, splitting of demands over equal *physical length* routes would likely produce a larger benefit than splitting equal *hop count* routes. This is because splitting over equal hop count routes could actually result in a higher overall installed bandwidth-distance product, and hence a higher design cost. Splitting is of use only in the realm of unit demand processing. When bundled demands are processed, they are generally meant to be carried intact on single routes, and not split across multiple systems.

A summary table of the main characteristics of the designs generated in Suite 1 is shown in Table 7.

Table 7: Test Suite 1 Results

	Network					
	netJ			netM		
	case23	case24	case25	case26	case27	case28
	Bundled/ 1-way/Hop	Unit/1-way/ Hop	Unit/k-way/ Hop	Bundled/ 1-way/Hop	Unit/1-way/ Hop	Unit/k-way/ Hop
Cost ($M)	5.18	4.71	4.86	411	371	370
Span cost ($M)	0.0920	0.0813	0.0838	383	344	344
Node cost ($M)	5.08	4.63	4.77	28.2	26.4	26.2
# Rings	11	10	11	27	26	26
Avg. # hops	5.45	5.50	5.18	4.89	4.73	4.73
Avg. circumference	41.8	40.7	38.1	1181.5	1104.0	1102.8
# ADMs	48	43	44	115	106	105
Avg. # ADMs/ring	4.36	4.30	4.00	4.26	4.08	4.04
Transitions	1674	1558	1636	1282	1198	1202
Working bandwidth-distance product	21871 DS1-km	21871 DS1-km	21966 DS1-km	779705 STS1-km	779705 STS1-km	786359 STS1-km
Margin bandwidth-distance product	55399 DS1-km	46433 DS1-km	48454 DS1-km	751543 STS1-km	598135 STS1-km	589945 STS1-km
Protection bandwidth-distance product	77270 DS1-km	68304 DS1-km	70420 DS1-km	1531248 STS1-km	1377840 STS1-km	1376304 STS1-km
Avg. specific progress	0.00324 DS1-km/$	0.00366 DS1-km/$	0.00333 DS1-km/$	0.000927 STS1-km/$	0.00100 STS1-km/$	0.00102 STS1-km/$
Avg. balance	0.206	0.211	0.202	0.202	0.221	0.225
Avg. capture	0.926	0.932	0.926	0.921	0.931	0.931

Test Suite 2: Optimization of Demand Routing

This test suite compares the use of shortest physical path routing versus shortest hop routing for netJ and netM. The test case grouping is summarized in Table 8 and the detailed summary of results is given in Table 9.

Table 8: Test Suite 2

Network	Routing Strategy	
	Shortest Physical Path Routing	Minimum Hop Count Routing
netJ	case29	case23
netM	case30	case26

Case29 shows that the use of real distance minimization is inferior to hop count minimization in netJ. The network design cost in case29 increases from $5.18M to $5.46M despite a decrease in the number of inter-ring transitions and installed working bandwidth-distance product. The number of ring systems increases from 11 to 14, and the number of required ADMs increases from 48 to 54. This is due to the fact that node costs dominate over line costs in the metropolitan context, and minimizing line costs causing an increase in the dominant cost component results in an increase in overall cost.

However, as might be expected, case20, the netM case, benefits from the use of shortest physical path routing. This is because in the long-haul context, line costs dominate over node costs and thus minimizing the dominant cost component, even to the detriment of the non-dominant one will result in a more efficient design. The design cost drops from $411M to $408M despite the increase in rings from 27 to 28, the increase of inter-ring transitions from 1282 to 1548, and the increase of margin bandwidth distance product from 751543 STS1-km to 811154 STS1-km. The installed working bandwidth distance product decreases from 779705 STS1-km to 702238 STS1-km, and the number of ADMs decreases from 115 to 113.

These two results confirm the widespread understanding among planners that in the metropolitan network context, hop count minimization is superior to demand route physical distance minimization. In the case of the long-haul network, the opposite is true; phys-

ical distance minimization is superior to hop count minimization.

Table 9: Test Suite2 Results

	Network			
	netJ		netM	
	case23	case29	case26	case30
	Bundled/1-way/Hop	Bundled/1-way/Real	Bundled/1-way/Hop	Bundled/1-way/Real
Cost ($M)	5.18	5.47	411	408
Span cost	0.0920	0.116	383	378
Node cost	5.08	5.35	28.2	29.3
# Rings	11	14	27	28
Avg. # hops	5.45	5.93	4.89	4.86
Avg. circumference	41.8	41.6	1181.5	1126.0
# ADMs	48	54	115	113
Avg. # ADMs/ring	4.36	3.86	4.26	4.04
Transitions	1674	1506	1282	1548
Working bandwidth-distance product	21871 DS1-km	21142	779705	702238
Margin bandwidth-distance product	55399 DS1-km	76591DS1-km	751543 STS1-km	811154 STS1-km
Protection band-width-distance product	77270 DS1-km	97733 DS1-km	1531248 STS1-km	1513392 STS1-km
Avg. specific progress	0.00324 DS1-km/$	0.00277 DS1-km/$	0.000927 STS1-km/$	0.000930 STS1-km/$
Avg. balance	0.206	0.118	0.202	0.201
Avg. capture	0.926	0.949	0.921	0.885

Test Suite 3: Balance vs. Capture Optimization versus Direct Cost

This test suite shows the relative performance of balance vs. capture optimization versus direct cost optimization in netJ and netM. Balance vs. capture optimization requires a sweep of the balance bias factor, alpha, to find a minimum cost design. It is not currently possible to know the optimum relative weighting of balance and capture prior to sweeping alpha, although in metropolitan cases it can fairly confidently be set to zero (full capture). In contrast, the direct cost optimization approach generates an optimized design in a single

run. Direct cost optimization also provides a framework to compare different ring technologies at each ring selection point, something that has not yet been implemented with balance vs. capture optimization.

Table 10 shows the test cases involved in the balance vs. capture optimization sweep and the baseline direct cost run to which they are compared.

Table 10: Test Suite 3

Network	Optimization Strategy		
	balance-capture		Direct Cost
	case	balance bias	
netJ	case31b00	0.0	case23
	case31b01	0.1	
	case31b02	0.2	
	case31b03	0.3	
	case31b04	0.4	
	case31b05	0.5	
	case31b06	0.6	
	case31b07	0.7	
	case31b08	0.8	
	case31b09	0.9	
	case31b10	1.0	
netM	case32b00	0.0	case26
	case32b01	0.1	
	case32b02	0.2	
	case32b03	0.3	
	case32b04	0.4	
	case32b05	0.5	
	case32b06	0.6	
	case32b07	0.7	
	case32b08	0.8	
	case32b09	0.9	
	case32b10	1.0	

Results for this set of experiments are shown in Table 11 and Table 12. In the balance vs. capture experiments, the best found netJ design occurs with an alpha value of 0.0, whereas the lowest cost design for netM occurs with an alpha value of 0.8. A plot of design cost vs. alpha for netJ is shown in Figure 50 and a corresponding plot for netM is shown in Figure 51. These plots illustrate the relative importance of capture for metropolitan and long-haul designs.

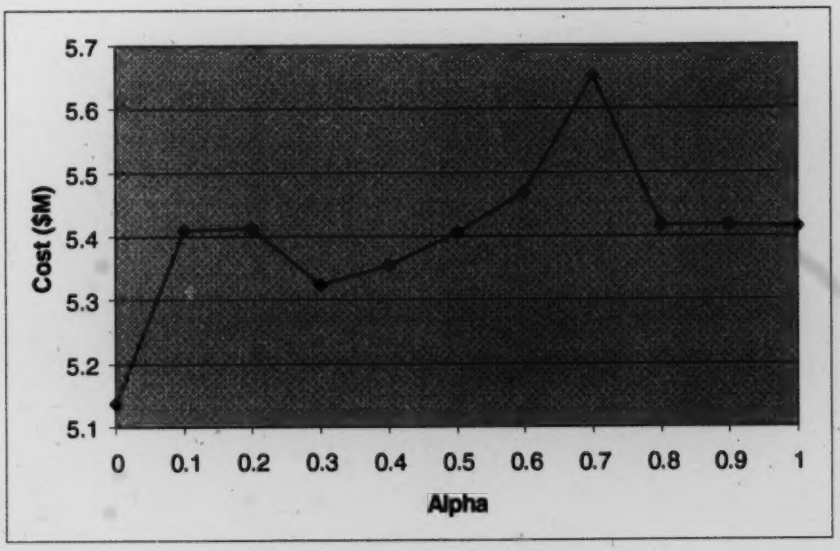


Figure 50: netJ -- Cost vs. Alpha

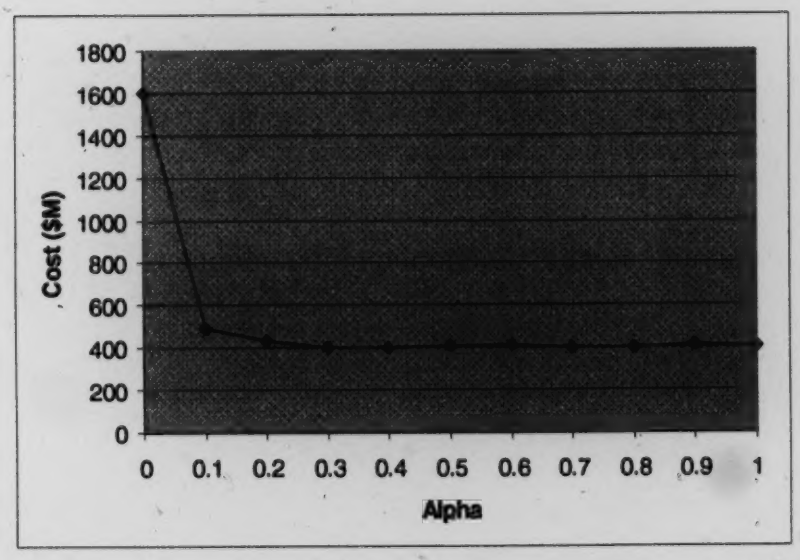


Figure 51: netM -- Cost vs. Alpha

As intended in the balance vs. capture design method, a higher capture bias tends to minimize the number of inter-ring transitions, thus minimizing the node cost. Node cost is more important than span cost in netJ. Therefore, minimizing the number of transitions

decreases the node cost and results in a lower cost design for netJ. A plot of the number of transitions vs. alpha is shown in Figure 52. The lowest cost design occurs at alpha = 0.0, corresponding to the minimum transition count.

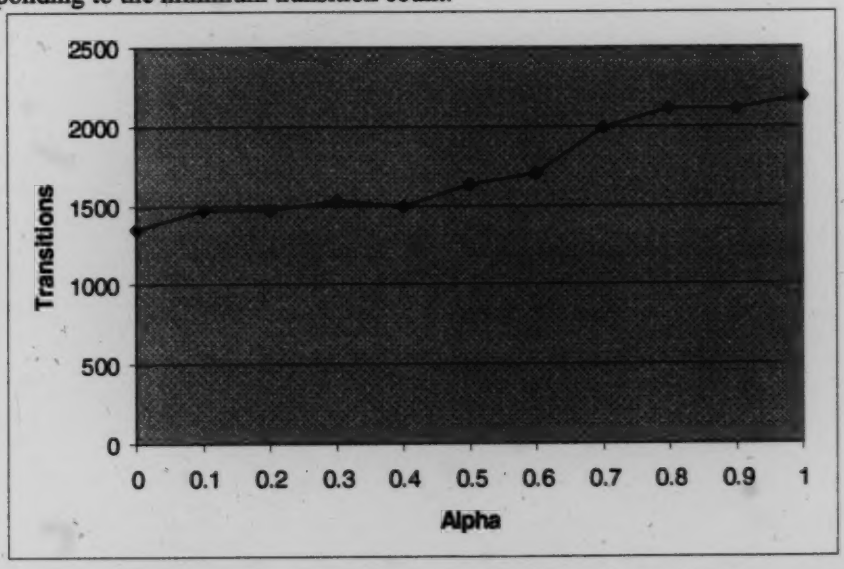


Figure 52: netJ – Number of Transitions vs. Alpha

Figure 53 shows that the minimum cost design for netJ occurs with the maximum installed protection bandwidth-distance product. Installed protection bandwidth-distance product is directly proportional to total installed bandwidth.

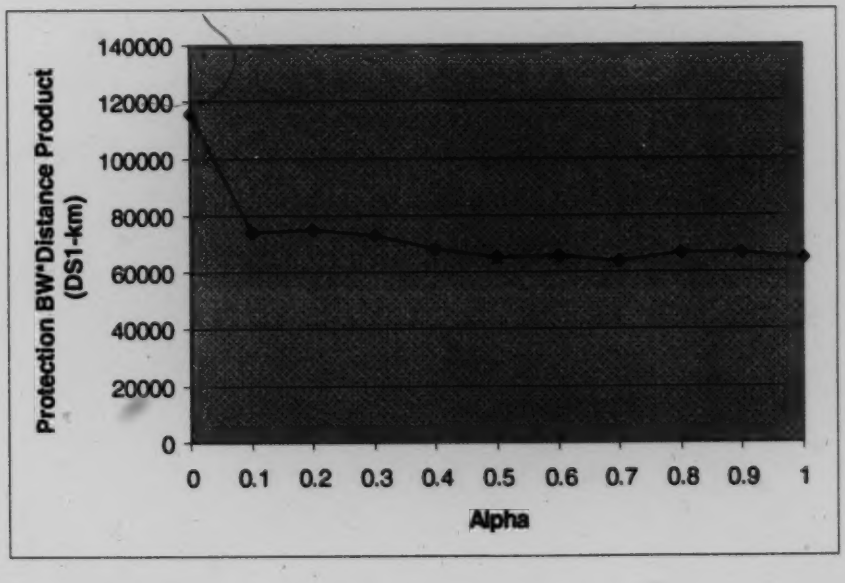


Figure 53: netJ -- Protection BW*Distance vs. Alpha

Thus, as expected in the long-haul designs, balance optimization is more important. High balance indicates higher span fill and better resource utilization. This is of prime importance in the long-haul network where span costs dominate node costs. A plot of protection bandwidth-distance product vs. alpha is shown in Figure 54. A plot of transition count vs. alpha for netM is shown in Figure 55. This figure demonstrates that the minimization of transition count is not a prime optimization objective in the long-haul design, as evidenced by the near maximum transition count of the minimum cost design at alpha = 0.8.

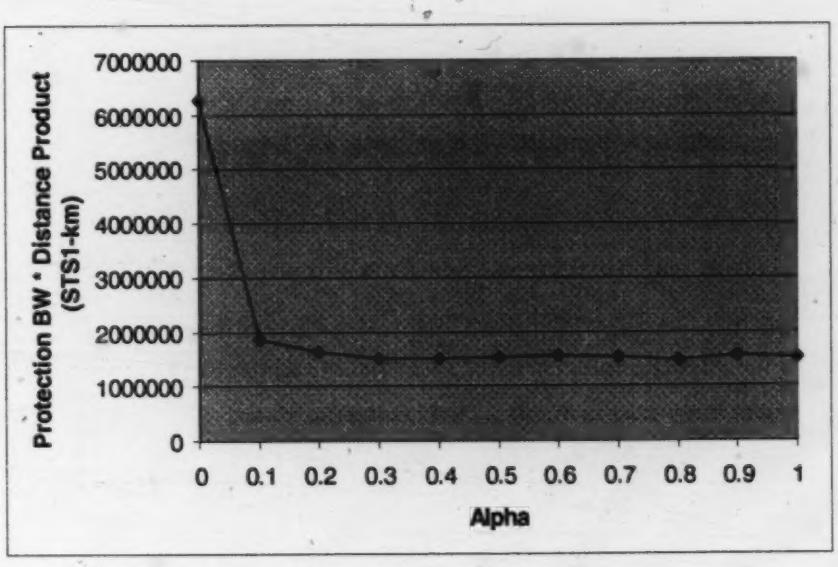


Figure 54: netM -- Protection BW*Distance vs. Alpha

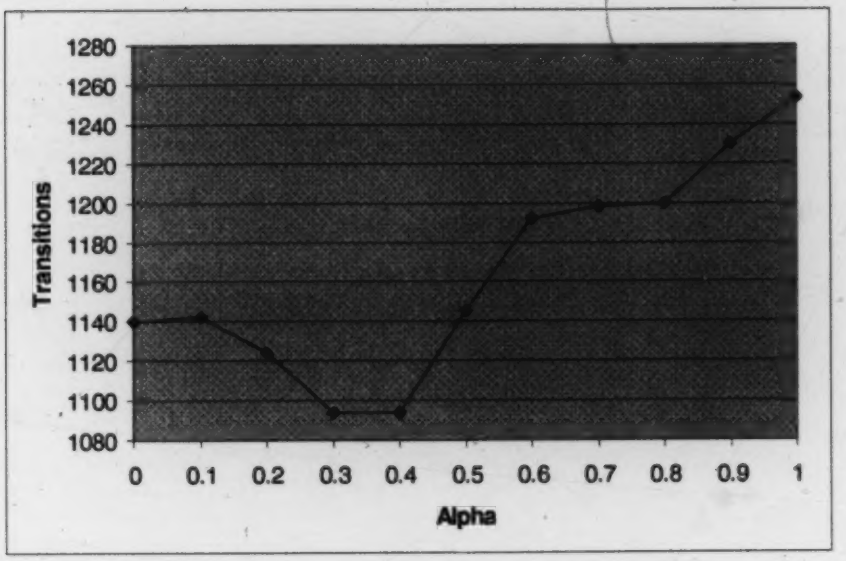


Figure 55: netM -- Number of Transitions vs. Alpha

The balance vs. capture optimized results for both netJ and netM are marginally better than their direct cost counterparts. However, it took 11 runs to find a superior result to the direct cost result in each case. This is an understandable finding. Even though it appears from first principles that the direct cost approach is an inherently better idea, the result it provides here is obtained from only one trial, whereas the alpha search method for balance vs. capture enjoys the benefits of many design trials from which the single best is selected. The best netJ design occurs at an alpha of 0, and has a cost of $5.14M. This is only 0.8% less expensive than the reference direct cost optimized design. The baseline design is not the best single technology direct cost design achieved.

In the netM study, the best balance vs. capture result occurs at an alpha of 0.8, with a corresponding cost of $399M, which is 2.9% less expensive than the reference design. However, the best balance vs. capture result is inferior to the best single technology direct cost optimization run (case27), which has a cost of $371M.

The best design results also have the highest average values of specific progress per ring placed. Specific progress can be used to rank the relative results so that the minimum cost design can be determined. Plots of specific progress vs. alpha are shown in Figure 56

for netJ and Figure 57 for netM. In both cases, the peak specific progress data point did correspond to the lowest cost design, confirming that the direct cost method is a well-founded principle.

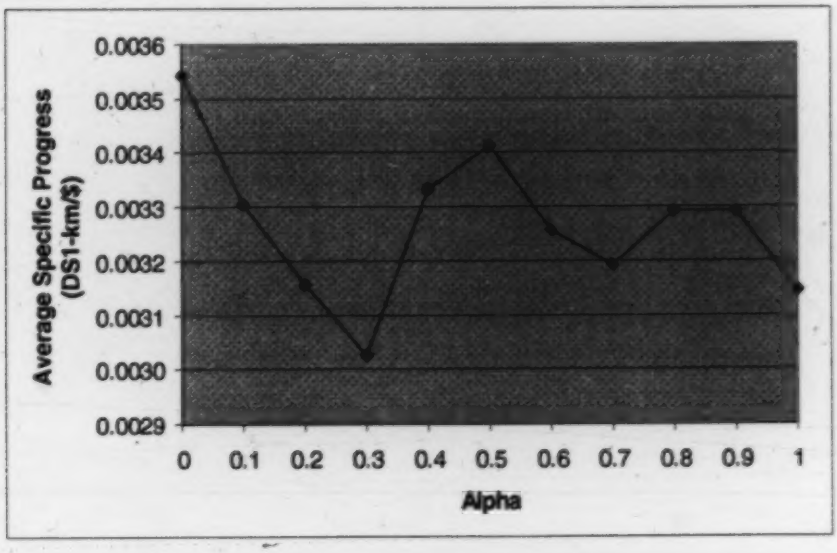


Figure 56: netJ -- Average Specific Progress vs. Alpha

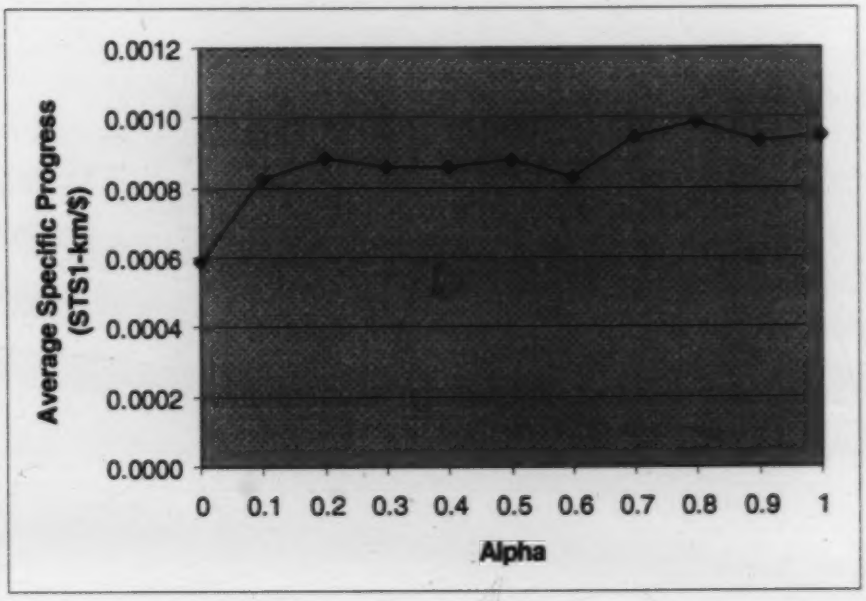


Figure 57: netM -- Average Specific Progress vs. Alpha

Figure 58 shows the that the minimum cost netJ design has a minimum number of ring systems. The same holds true for netM, as shown in Figure 59. Notably, however, the minimum cost designs do not have the minimum number of ADM terminals, as shown in Figure 60 for netJ and in Figure 61 for netM.

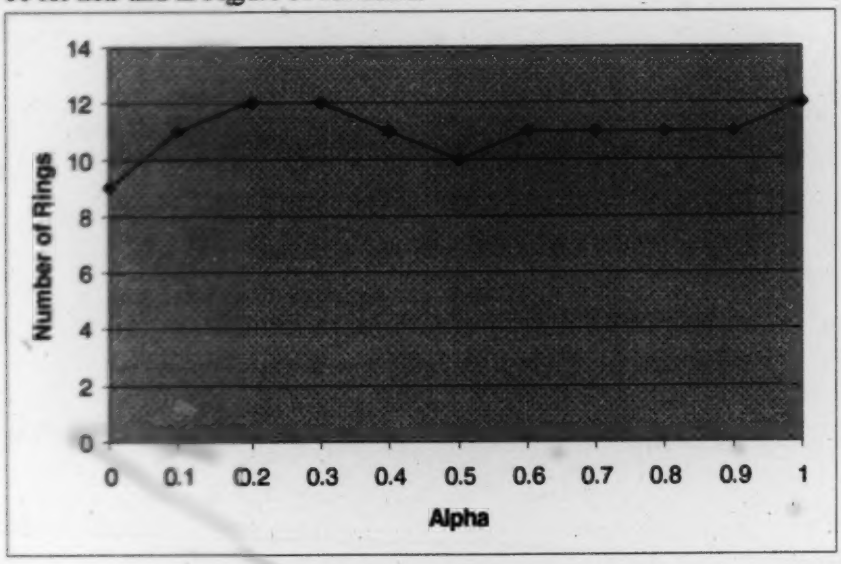


Figure 58: netJ – Number of Rings vs. Alpha

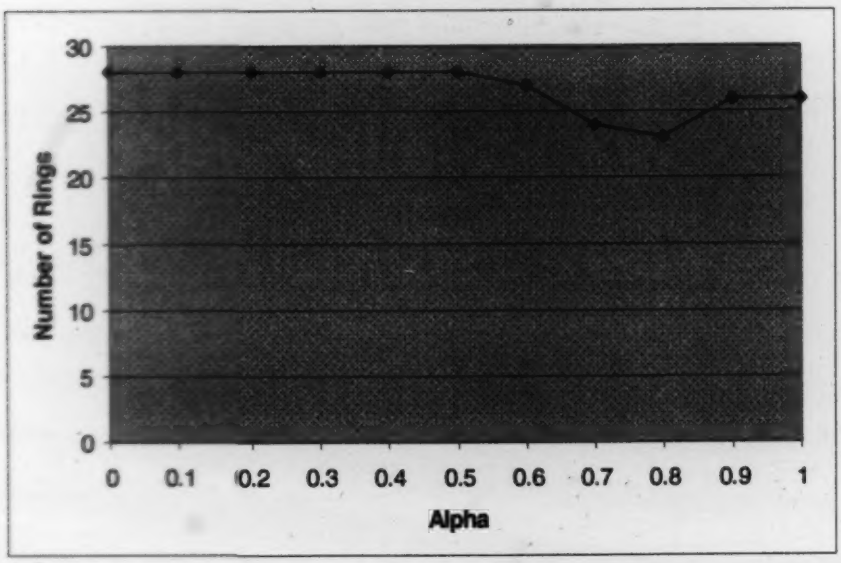


Figure 59: netM – Number of Rings vs. Alpha

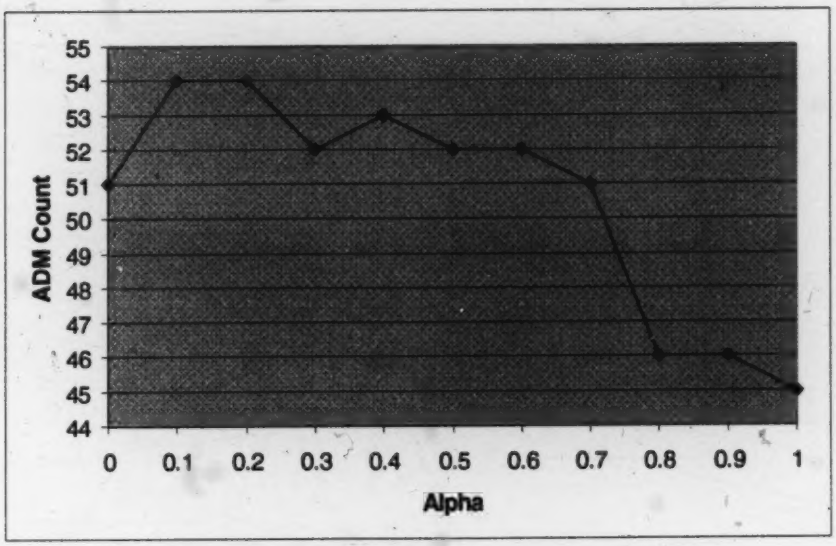


Figure 60: netJ – ADM Count vs. Alpha

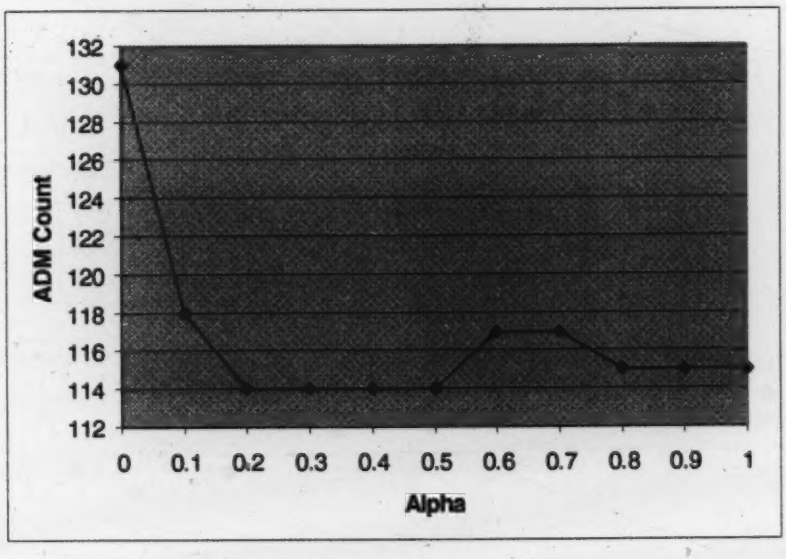


Figure 61: netM – ADM Count vs. Alpha

The lowest cost netJ design has the best average capture, but not the least installed bandwidth. The lowest cost netM design in this test suite has the highest average balance.

In contrast, the lowest cost netM design does have minimum installed bandwidth. The least expensive netJ design has the lowest node cost, whereas the least expensive netM design has the lowest span or installed bandwidth cost. Plots of total, span, and node costs vs. alpha are shown in Figure 62 for netJ, and Figure 63 for netM. These results demonstrate that node costs dominate in the metropolitan network context, whereas span costs dominate in the long-haul network context.

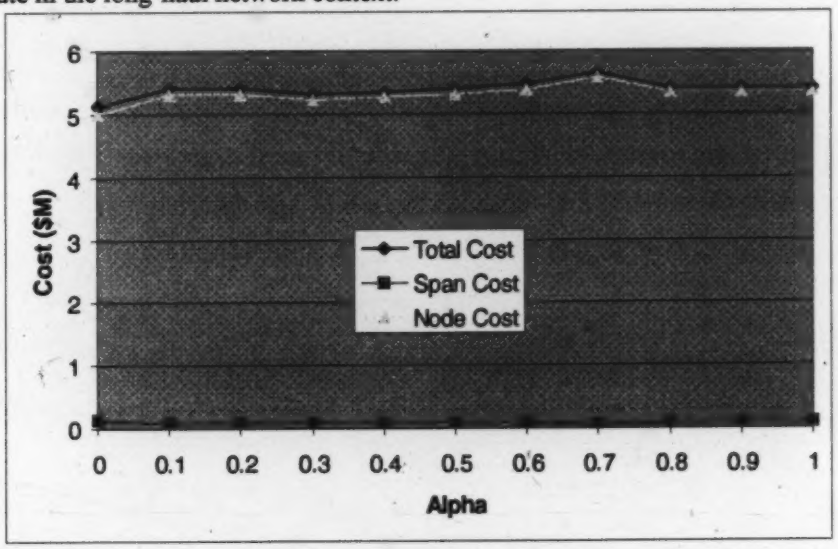


Figure 62: netJ – Total Cost, Span Cost and Node Cost vs. Alpha

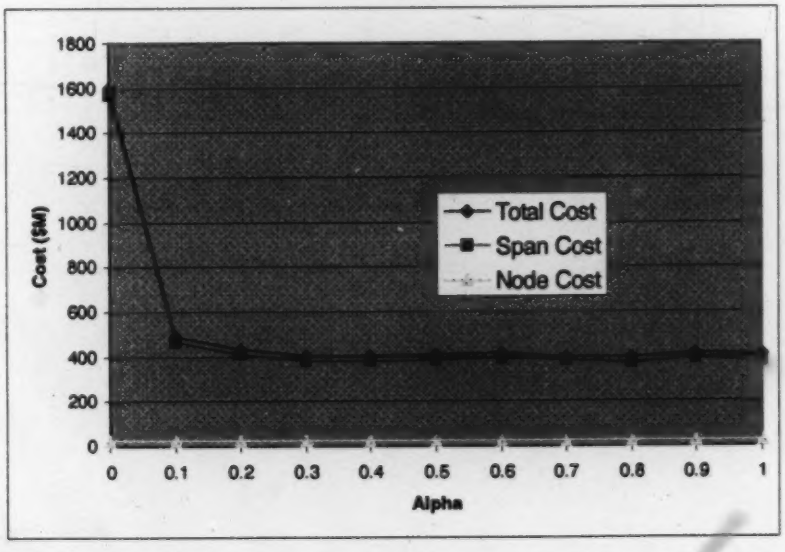


Figure 63: netM – Total Cost, Span Cost, and Node Cost vs. Alpha

Figure 64 and Figure 65 show the opposing nature of balance and capture. Optimizing for capture by minimizing node costs results in non-optimal balance and non-minimal span costs. Optimizing for balance by minimizing span costs results in non-optimal capture and hence excess node costs. The unexpected behavior of the capture figure of merit in the netM test case is due to the greedy individual ring choices actually causing an inferior overall design. The pure capture design was an outlier, with a cost almost 3 times higher than any other balance vs. capture design generated for that network.

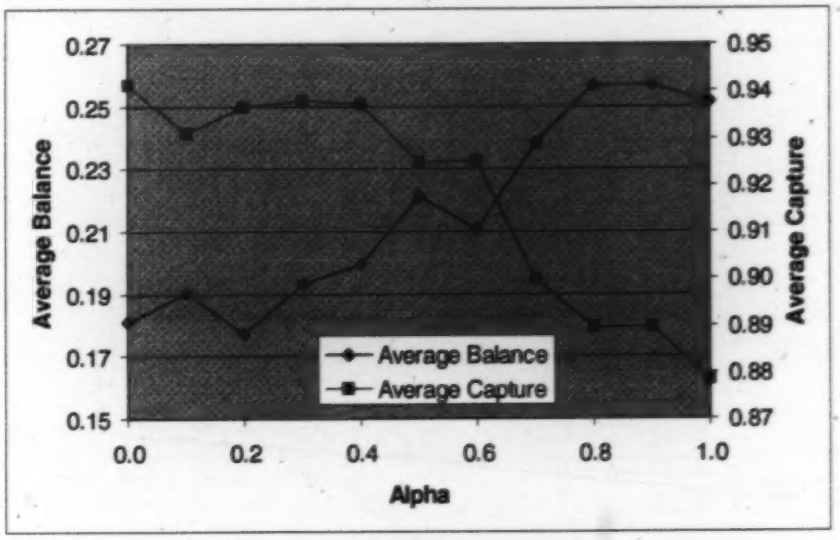


Figure 64: netJ -- Average Balance and Capture vs. Alpha

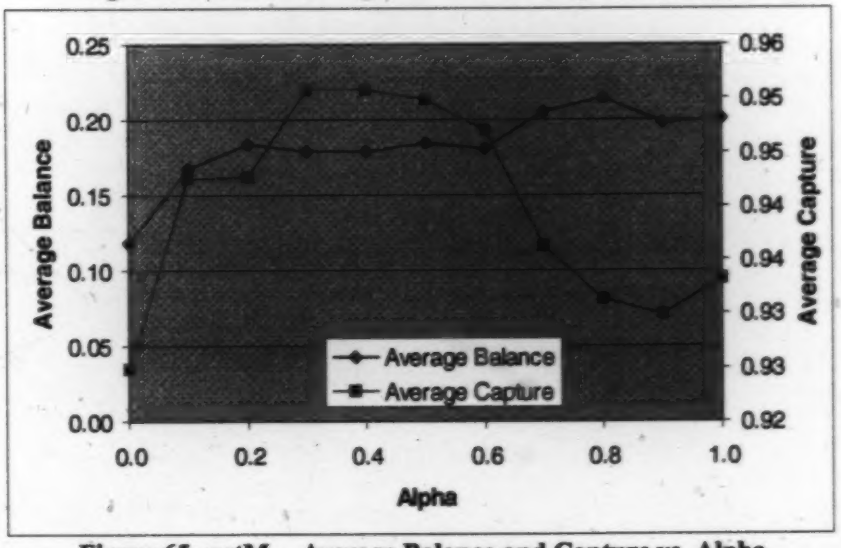


Figure 65: netM -- Average Balance and Capture vs. Alpha

Table 11: Test Suite 3 Results for netJ

Network	case23	case31										
netJ	Direct Cost	Bal/Cap										
alpha		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Cost ($M)	5.18	5.14	5.41	5.41	5.32	5.35	5.41	5.47	5.65	5.42	5.42	5.41
Span cost ($M)	0.0920	0.138	0.0881	0.0892	0.0869	0.0814	77883.2	78177	75883.2	79350.4	79350.4	77213.6
Node cost ($M)	5.08	5.00	5.32	5.32	5.24	5.27	5.33	5.39	5.58	5.34	5.34	5.34
# Rings	11	9	11	12	12	11	10	11	11	11	11	12
Avg. # hops	5.45	9.89	5.73	5.25	5.25	5.18	5.50	5.00	5.00	5.09	5.09	4.50
Avg. circumference	41.8	76.5	40.0	37.2	36.2	37.0	38.9	35.5	34.5	36.1	36.1	32.2
# ADMS	48	51	54	54	52	53	52	52	51	46	46	45
Avg. # ADMs/ring	4.36	5.67	4.91	4.50	4.33	4.82	5.20	4.73	4.64	4.18	4.18	3.75
Transitions	1674	1346	1474	1474	1534	1496	1636	1704	1988	2112	2112	2190
Working bandwidth-distance product (DS1-km)	21871	21871	21871	21871	21871	21871	21871	21871	21871	21871	21871	21871
Margin bandwidth-distance product (DS1-km)	55399	93839	52121	53036	51096	46508	43551	43798	41871	44784	44784	42989
Protection bandwidth-distance product (DS1-km)	77270	115710	73992	74907	72967	68379	65429	65669	63742	66654	66654	64859
Avg specific progress (DS1-km/$)	0.00324	0.00354	0.00331	0.00316	0.00303	0.00333	0.00341	0.00326	0.00319	0.00329	0.00329	0.00315
Avg. balance	0.206	0.181	0.190	0.177	0.193	0.199	0.221	0.211	0.239	0.257	0.257	0.252
Avg. capture	0.926	0.941	0.931	0.937	0.938	0.937	0.925	0.925	0.900	0.890	0.890	0.878
Avg. bal/cap		0.941	0.857	0.785	0.715	0.642	0.573	0.497	0.437	0.383	0.320	0.252

Table 12: Test Suite 3 Results for netM

Network	case26	case32										
netM	Direct Cost	Bal/Cap										
alpha		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Cost ($M)	411	1596	492	436	405	405	411	418	399	399	415	408
Span cost ($M)	383	1569	466	411	380	380	386	392	385	374	389	383
Node cost ($M)	28.2	27.5	25.6	25.0	24.9	24.9	25.1	25.8	25.8	25.5	25.7	25.8
# Rings	27	28	28	28	28	28	28	27	24	25	26	26
Avg. # hops	8.89	16.1	5.61	5.36	5.00	5.00	5.14	5.37	5.58	5.61	5.15	5.15
Avg. circumference	1181.5	4669.2	1388.3	1222.5	1130.2	1130.2	1148.0	1209.3	1335.3	1354.3	1246.7	1226.0
# ADMS	115	131	118	114	114	114	114	117	117	115	115	115
Avg. # ADMs/ring	4.26	4.68	4.21	4.07	4.07	4.07	4.07	4.33	4.88	5.00	4.42	4.42
Transitions	1282	1140	1142	1124	1094	1094	1146	1192	1198	1200	1230	1254
Working bandwidth-distance product (STS1-km)	779705	779705	779705	779705	779705	779705	779705	779705	779705	779705	779705	779705
Margin bandwidth-distance product (STS1-km)	751543	5495719	1086103	863287	739255	739255	763255	787591	758599	715447	776215	1530000
Protection bandwidth-distance product (STS1-km)	1531248	6275424	1865808	1642992	1518960	1518960	1542960	1567296	1538304	1495132	1555920	1530000
Avg specific progress (STS1-km/$)	0.000927	0.000588	0.000825	0.000886	0.000861	0.000861	0.000883	0.000831	0.000944	0.000990	0.000931	0.000949
Avg. balance	0.202	0.118	0.168	0.184	0.179	0.179	0.185	0.181	0.205	0.215	0.198	0.201
Avg. capture	0.921	0.925	0.942	0.943	0.951	0.951	0.950	0.947	0.936	0.931	0.930	0.933
Avg. bal/cap		0.925	0.865	0.791	0.719	0.642	0.567	0.487	0.424	0.358	0.272	0.201

Test Suite 4: Ring Candidate Loading Method

This test suite examines the advantages of a complex, multistage pre-loading demand sorting strategy compared to a simple route selection strategy for loading. The simple loader arranges the demand routes in order of decreasing bandwidth-distance product based on their length of travel on the cycle being demand loaded.

The complex loading strategy attempts to optimize the ring loading process by sorting the demand routes in a more sophisticated manner. The demand routes are ordered first by decreasing value of the ratio of links in common with the current cycle to the total number of unrouted links remaining for that demand route. Any ties are broken by sorting in order of decreasing bandwidth distance product. Routes which have the same bandwidth distance product are differentiated by giving higher priority to the routes with the longer physical distance of travel.

The test cases comprising Test Suite 4 are shown in Table 13.

Table 13: Test Suite 4

Network	Pre Loading Route Sorting Strategy	
	Simple	Complex
netJ	case23	case33
netM	case26	case34

The results of the experiments are shown in Table 14. The lowest cost design has both the best average specific progress, and the best average balance. The netJ design produced with the complex route sorter is superior to the baseline case produced with the simple route sorter, resulting in a design cost of $4.71M, 9.0% less expensive than the baseline design. The cost of the netM design produced with the complex route sorter was $436M, which is 6.2% more expensive than the baseline design. The complex route sorter attempts to optimize fill and hence balance, and may thus adversely affect capture in the netJ case. It results in a superior design, nonetheless, to the decrease in span and node costs. The complex route sorter actually improves capture in the netM case, but decreases the average balance, indicating lower overall system fill. As well, the number of fibre systems, fibre mileage, and the attendant number of ADMs increases, resulting in higher total

span cost. The overall system cost increases with the use of the complex route sorter due to the dominance of span cost in the long-haul network.

Table 14: Test Suite 4 Results

	netJ		netM	
	case23	case33	case26	case34
	Simple Route Sort	Complex Route Sort	Simple Route Sort	Complex Route Sort
Cost ($M)	5.18	4.71	411	436
Span cost ($M)	0.0920	0.0719	383	410
Node cost ($M)	5.08	4.64	28.2	26.1
# Rings	11	8	27	29
Avg. # hops	5.45	6.13	4.89	4.86
Avg. circumference	41.8	45.0	1181.5	1179.2
# ADMs	48	43	115	118
Avg. # ADMs/ring	4.36	5.38	4.26	4.07
Transitions	1674	1570	1282	1228
Working bandwidth-distance product	21871 DS1-km	21871 DS1-km	779705 STS1-km	779705 STS1-km
Margin bandwidth-distance product	55399 DS1-km	38547DS1-km	751543 STS1-km	861703 STS1-km
Protection bandwidth-distance product	77270 DS1-km	60418 DS1-km	1531248 STS1-km	1641408 STS1-km
Avg. specific progress	0.00324 DS1-km/$	0.00385 DS1-km/$	0.000927 STS1-km/$	0.000919 STS1-km/$
Avg. balance	0.206	0.266	0.202	0.200
Avg. capture	0.926	0.924	0.921	0.924

Test Suite 5: Node Constraints

This test suite shows how the ring designs are modified by disallowing inter-ring transiting at a particular node. Such constraints exist where DCS machines or crossconnect panels are not available at a node and traffic cannot be exchanged between rings.

The test cases are outlined in Table 15, where the constrained case35 and case36 are

compared with the unconstrained baseline case23 and case26, respectively.

Table 15: Test Suite 5

Network	Network Constraints	
	None	Transition Constrained
netJ	case23	case35
netM	case26	case36

The results of these experiments are shown in Table 16. As would be expected, constraining the design by disallowing transitions at a node increases the design cost in both the netJ and the netM cases. The netJ design cost increases by 7.8% to $5.58M and the netM design cost increases by 21.2% to $498M when inter-ring transitions are disallowed at one node. Both the node and span costs increase in the constrained test case relative to the baseline case for netJ. The node cost increases by 7.5% to $5.47M, and the span cost increases by 23.6% to $114k.

Constraining transitions in the netM case lowers the node cost because the transition count is lowered. However, the span cost increases more than proportionately for added routing to the allowed transition points. The node cost decreases by 5.8% to $26.6M whereas the span cost increases by 23.2% to $472M. In both cases, the lowest cost design has the best average balance and the best average specific progress.

Table 16: Test Suite 5 Results

	Network			
	netJ		netM	
	case23	case35	case26	case36
	Unconstrained	Transition Constrained	Unconstrained	Transition Constrained
Cost ($M)	5.18	5.58	411	498
Span cost ($M)	0.0920	0.114	383	472
Node cost ($M)	5.08	5.47	28.2	26.6
# Rings	11	13	27	27
Avg. # hops	5.45	6.08	4.89	5.26
Avg. circumference	41.8	43.7	1181.5	1455.9
# ADMs	48	56	115	122
Avg. # ADMs/ring	4.36	4.31	4.26	4.52
Transitions	1674	1478	1282	1216
Working bandwidth-distance product	21871 DS1-km	21871 DS1-km	779705 STS1-km	779705 STS1-km
Margin bandwidth-distance product	55399 DS1-km	73644 DS1-km	751543 STS1-km	1107079 STS1-km
Protection bandwidth-distance product	77270 DS1-km	95515 DS1-km	1531248 STS1-km	1886784 STS1-km
Avg. specific progress	0.00324 DS1-km/$	0.00293 DS1-km/$	0.000927 STS1-km/$	0.000897 STS1-km/$
Avg. balance	0.206	0.165	0.202	0.196
Avg. capture	0.926	0.944	0.921	0.933

Test Suite 6: Add-Drop Constraints on ADMs

This test suite examines the effect of applying two types of ADM constraints to generate metropolitan and long-haul network designs. The constituent test cases comprising test suite 6 are shown in Table 17. For both netJ and netM, the unconstrained baseline cases are compared to ADM port count constrained and active ADM count constrained test cases. ADM port count constraints are used to model real systems where only a subset of the line bandwidth can be add/dropped at any node. Such constraints occur in ADMs because of mechanical shelf space limitations imposed by physical design parameters. The ADM port count constraint applied in this test suite limits the amount of add-drop band-

width to one-half of the aggregate line bandwidth of the system.

The number of active ADMs in a SONET ring system is restricted to 16 because of the addressing limitations of the SONET standard. The baseline test case is synthesized with this limitation in place. The active ADM count constraint in this test suite further limits the active number of ADMs to 2. This particular constraint provides a convenient way to model diversely routed 1:1 linear systems within the iterative ring design framework. For instance, in a multiple-technology design, RingBuilder™ could be given a standard OC-48 BLSR system and an OC-48 BLSR with a 2 active node limit as two technology options to evaluate

Table 17: Test Suite 6

Network	ADM Constraint			
	Baseline (16 active ADMs, no port limits)	ADM Port Count Constraint	Active ADM Count Constraint	
			capture-biased	balance-biased
netJ	case23	case37	case38	not applicable
netM	case26	case39	case40	case40a

A summary of the results obtained in the test suite 6 test cases is shown in Table 18. Restricting the ADM port count for netJ raises the total design cost by 16.8%. Both the span and node costs increase and the number of ring systems increase from 11 to 13. The reason is clear: to achieve the required add/drop quantities, whole additional rings have to be commissioned. The number of ADMs increases from 48 to 63, and the total number of transitions decrease from 1674 to 1456. Systems that limit the amount of add/drop bandwidth at any one node can therefore greatly increase the cost of deploying a ring network because of their lower bandwidth efficiency.

The application of the active ADM count restrictions to the extreme limiting case of 2, increases the cost by 91.1%. The span and node costs increase as the number of ring systems installed increases from 11 to 56. Clearly, in the metropolitan context, the use of point-to-point systems in lieu of rings is extremely inefficient.

For netM, restricting the ADM port count raises the network design cost by 12.8%. Both the node and span costs increase, and the number of ring systems also increases from

27 to 31 systems. The number of ADMs increases from 115 to 129, and the number of inter-ring transitions increases from 1282 to 1744. Limiting the add/drop bandwidth results in a less efficient network design. Limiting the active ADM count per system to two in the long haul case produces some interesting results. Using the baseline capture-biased demand route loading strategy in conjunction with the active ADM count restriction produces an incomplete design because unrouted demand segments are left at the termination of the design process. This is a limitation of the capture-biased loader, but this limitation only appears under extreme conditions such described below. However, the cost of even the incomplete design is 302.3% higher than the baseline design. Both the span and node costs increase relative to the baseline design, the number of ring systems increase from 27 to 106, and the number of ADMs increases from 115 to 216.

The design is incomplete because capture loading requires that *all* unrouted spans of a demand intersecting a cycle must be loaded by that cycle or that demand is left unrouted. If a demand is partially loaded onto a ring, with some route segments unrouted, and if those remaining segments can be routed on the same cycle, the capture biased loader in conjunction with the 2 ADM limit will reject the loading of the route segments onto the candidate cycle. This is illustrated in Figure 66.

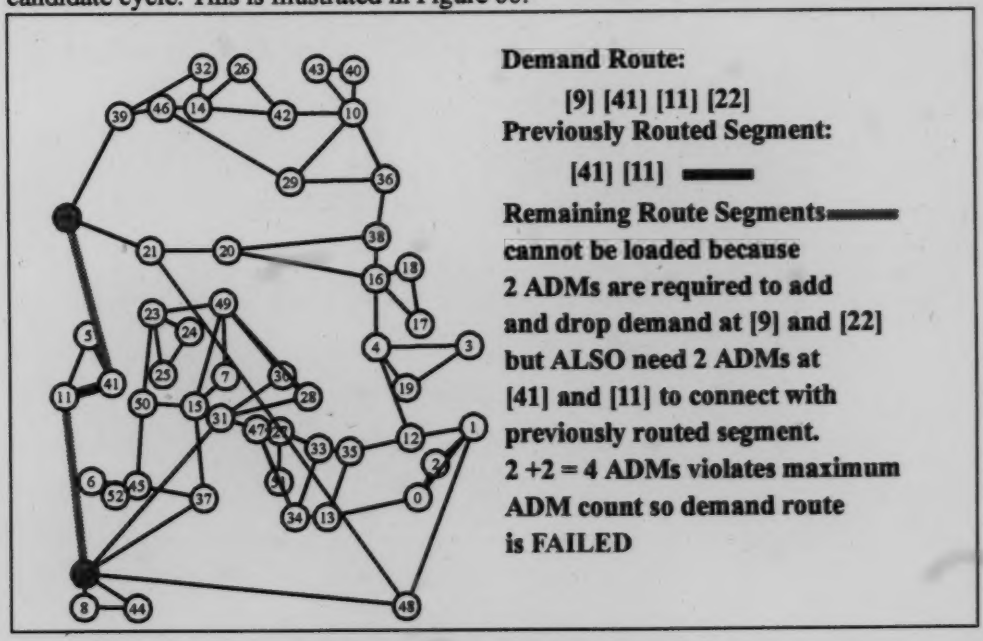

Demand Route:
[9] [41] [11] [22]
Previously Routed Segment:
[41] [11]
Remaining Route Segments
cannot be loaded because
2 ADMs are required to add
and drop demand at [9] and [22]
but ALSO need 2 ADMs at
[41] and [11] to connect with
previously routed segment.
2 +2 = 4 ADMs violates maximum
ADM count so demand route
is FAILED

Figure 66: Sample Failed Demand

Figure 67 shows all of the spans with unrouted demand segments highlighted, as well as the endpoints of the remaining unrouted demands at the termination of the design run.

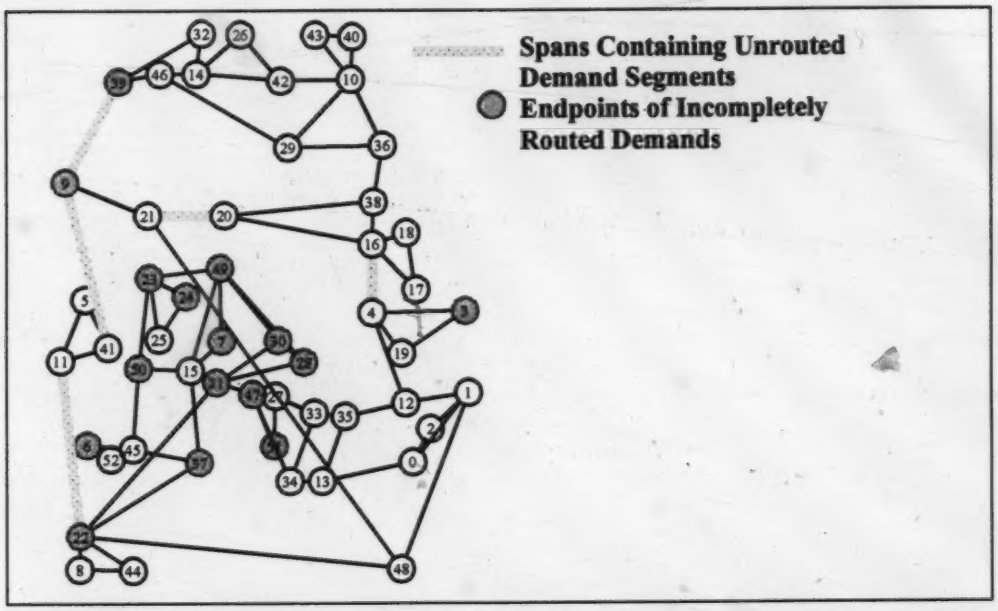


Figure 67: Demand Endpoints and Spans Containing Unrouted Demand Segments

Case40a, in which balance-biased demand loading is used in place of capture-biased demand loading, eliminates this problem. Balance-biased loading will load any segments of a demand which intersect the cycle being loaded, rather than insisting that all segments in common be loaded. The resulting design is very expensive, costing 189% more than the baseline case. The design of case40a consists of 90 rings and 180 ADMs. The number of inter-ring transitions is 2060, 60.7% higher than the baseline case. Just as in netJ, the exclusive use of 1:1 diversely routed systems to form a network design is very inefficient.

Again, the lowest cost netJ design has the highest average specific progress and the highest average balance. The lowest cost netM design also has the highest average specific progress, the highest average capture, and the highest average balance.

Table 18: Test Suite 6 Results

	Network						
	netJ			netM			
	case23	case37	case38	case26	case39	case40	case40a
						capture-biased	balance-biased
	Un-con-strained	ADM Port Count	Active ADM Count	Uncon-strained	ADM Port Count	Active ADM Count	Active ADM Count
Cost ($M)	5.18	6.04	9.89	411	464	1653	1189
Span cost ($M)	0.0920	0.106	0.429	383	436	1612	1150
Node cost ($M)	5.08	5.94	9.46	28.2	28.1	42.2	38.9
# Rings	11	13	56	27	31	106	90
Avg. # hops	5.45	6.23	5.81	4.89	5.03	5.24	4.86
Avg. circumference	41.8	40.9	37.6	1181.5	1171.2	1243.6	1065.0
# ADMs	48	63	114	115	129	216	180
Avg. # ADMs/ring	4.36	4.85	2.00	4.26	4.16	2.00	2.00
Transitions	1674	1456	1404	1282	1744	1716	2060
Working bandwidth-distance product	21871 DS1-km	21871 DS1-km	21871 DS1-km	779705 STS1-km	779705 STS1-km	766740 STS1-km	779705 STS1-km
Margin bandwidth-distance product	55399 DS1-km	67508 DS1-km	338431 DS1-km	751543 STS1-km	963079 STS1-km	5679996 STS1-km	3821287 STS1-km
Protection bandwidth-distance product	77270 DS1-km	89379 DS1-km	360302 DS1-km	1531248 STS1-km	1742784 STS1-km	6446736 STS1-km	4600992 STS1-km
Avg. specific progress	0.00324 DS1-km/$	0.00295 DS1-km/$	0.00201 DS1-km/$	0.000927 STS1-km/$	0.000870 STS1-km/$	0.000387 STS1-km/$	0.000474 STS1-km/$
Avg. balance	0.206	0.141	0.0389	0.202	0.171	0.0609	0.0760
Avg. capture	0.926	0.944	0.919	0.921	0.892	0.837	0.828
Comments						incomplete design	

Test Suite 7: Balance versus Capture Bias in Loading

This test suite examines the application of balance biased loading in place of capture biased loading in both the long-haul and metropolitan network context. The expectation is that capture-biased loading (which is intended to minimize inter-ring transiting on a per ring basis) is the option of choice for metropolitan networks, whereas balance-biased load-

ing should be preferred for long-haul networks, where high line capacity utilization is the most important. The individual test cases comprising test suite 7 are listed in Table 19. The capture biased router may not provide the overall lowest possible transition count. It minimizes the transition count on a per ring rather than on an overall basis.

Table 19: Test Suite 7

Network	Route Loading Strategy	
	Capture biased loading	Balance biased loading
netJ	case23	case41
netM	case26	case42

The results are shown in Table 20 where the balance-biased demand loading approach is found to be superior for both netJ and netM. The total cost, the span cost, and the node cost are lower for both networks. The total cost of the balance-biased demand loading design is $4.94M, which is 4.7% lower than the baseline capture-biased route loading design. The balance biased router designs have the same number of rings as their baseline counterparts (11 for netJ, 27 for netM), but the number of ADMs decreases. The average number of inter-ring transitions is higher. The lower number of ADMs coupled with the larger transition counts indicates that higher ring utilization is achieved at the expense of increased transition counts between the ring systems. The overall installed bandwidth decreases in both netJ and netM, as evidenced by the installed protection capacity.

The lowest cost netJ design has the highest average capture and the highest average specific progress. The least expensive netM design has the highest average balance, the highest average capture and the highest specific progress.

Table 20: Test Suite 7 Results

	Network			
	netJ		netM	
	case23	case41	case26	case42
	Capture-Biased	Balance-Biased	Capture-Biased	Balance-Biased
Cost ($M)	5..18	4.94	411	371
Span cost ($M)	0..0920	0.0774	383	346
Node cost ($M)	5..08	4.87	28.2	25.0
# Rings	11	11	27	27
Avg. # hops	5..45	4.91	4.89	4.67
Avg. circumference	41.8	35.2	1181.5	1067.9
# ADMs	48	44	115	106
Avg. # ADMs/ring	4..36	4.00	4.26	3.93
Transitions	1674	1744	1282	1744
Working bandwidth-distance product	21871 DS1-km	21871 DS1-km	779705 STS1-km	779705 STS1-km
Margin bandwidth-distance product	5:5399 DS1-km	43104 DS1-km	751543 STS1-km	604231 STS1-km
Protection bandwidth-distance product	77270 DS1-km	64975 DS1-km	1531248 STS1-km	1383936 STS1-km
Avg. specific progress	0..00324 DS1-km/$	0.00327 DS1-km/$	0.000927 STS1-km/$	0.000986 STS1-km/$
Avg. balance	0..206	0.204	0.202	0.215
Avg. capture	0..926	0.928	0.921	0.925

Test Suite 8: Multiple Ring Technologies

Suite 8 shows the effects of allowing multiple technology designs. Multiple technology designs are expected to be more cost-effective due to the more efficient use of network resources. Figure 21 outlines the test cases in test suite 8, which compares the single-technology baseline designs with multiple technology runs.

Table 21: Test Suite 8

Network	Design Technology	
	Single Technology Optimization	Multiple Technology Optimization
netJ	case23	case43
netM	case26	case44

The technology choices afforded to RingBuilder™ for the netJ design in case43 are shown in Table 22. The technology choices for netM in case44 are shown in Table 23.

Table 22: Technology Choices for Multi-Technology netJ Design

Technology	System Capacity			System Cost Components			
	Demand Management Unit	Max # ADMs	Max ADM Add/ Drops	ADM Common cost ($)	ADM Add/Drop Cost ($)	ADM Transit Cost ($)	Fibre Cost ($)
2BLSR12	DS1	16	336	70000	70	900	100
2BLSR12	DS1	16	1344	140000	70	900	100
2UPSR12	DS1	16	336	60000	70	900	100
4BLSR12	DS1	16	672	100000	70	900	100

Table 23: Technology Choices for Multi-Technology netM Design

Technology	System Capacity			System Cost Components			
	Demand Management Unit	Max # ADMs	Max ADM Add/ Drops	ADM Common cost ($)	ADM Add/ Drop Cost ($)	ADM Transit Cost ($)	Fibre Cost ($)
4BLSR48	STS1	16	96	160000	2000	5000	3000
2BLSR192	STS1	16	192	320000	2000	5000	3000
2UPSR192	STS1	16	192	240000	2000	5000	3000
2UPSR48	STS1	16	48	160000	2000	5000	3000
2UPSR48	STS1	2	48	100000	2000	5000	3000
2UPSR192	STS1	2	192	220000	2000	5000	3000

Case23, the netJ baseline test case, is comprised of 11 2BLSR12 rings. In contrast, the multi-technology design generated in case43 is made up of twelve 2UPSR12 systems and one 2BLSR12 system, and the multi-technology design of case44 consists of one 2BLSR192, four 2UPSR192 and thirty-one 2UPSR48 systems. Total cost and span cost decrease in both the netJ and the netM test cases through the use of multiple technologies. The netJ design drops in cost by 3.4% to $5.00M, whereas the netM design drops by 58.0% to $173M. The node cost increases by 4.53% in the netM case, but span cost decreases substantially, by 62.6%. The number of installed systems increases in both networks in the multiple technology designs, with the ADM count increasing in the netJ design but not in the netM design. The amount of installed margin also increases in the netJ design, indicating that the multiple technology network design is not only less expensive to implement, but also potentially capable of handling more demand growth if the increased demand is in the part of the network where the margin exists.

The lowest cost netJ design has the highest average specific progress. The lowest cost netM design has the highest average capture and the highest average specific progress.

Table 24: Test Suite 8 Results

	Network			
	netJ		netM	
	case23	case43	case26	case44
	Single Technology	Multiple Technology	Single Technology	Multiple Technology
Cost ($M)	5.18	5.00	411	173
Span cost ($M)	0.0920	0.0888	383	143
Node cost ($M)	5.08	4.91	28.2	29.5
# Rings	11	13	27	36
Avg. # hops	5.45	4.69	4.89	4.17
Avg. circumference	41.8	34.2	1181.5	662.5
# ADMs	48	53	115	115
Avg. # ADMs/ring	4.36	4.08	4.26	3.19
Transitions	1674	1628	1282	1274
Working bandwidth-distance product	21871 DS1-km	21871 DS1-km	779705 STS1-km	779705 STS1-km
Margin bandwidth- distance product	55399 DS1-km	64505 DS1-km	751543 STS1-km	618860 STS1-km
Protection bandwidth-distance product	77270 DS1-km	143252 DS1-km	1531248 STS1-km	2542608 STS1-km
Avg. specific progress	0.00324 DS1-km/$	0.00364 DS1-km/$	0.000927 STS1-km/$	0.00127 STS1-km/$
Avg. balance	0.206	0.137	0.202	0.105
Avg. capture	0.926	0.900	0.921	0.940
# $2BLSR12_{DS1largest}$	11	1		
# $2UPSR12_{DS1largest}$		12		
# $4BLSR48_{STS1largest}$			27	
# $2UPSR48_{STS1largest}$				31
# $2BLSR192_{STS1largest}$				1
# $2UPSR192_{STS1largest}$				4

Test Suite 9: Best Option Designs

Design suite 9 ties together the best design choices of all of the previous test cases and generates two 'ultimate' designs, comparing them to the baseline design. The test cases comprising suite 9 are shown in Table 25.

Table 25: Test Suite 9

Network	Base Design	Best Features Design		
		Based on 'experience and intuition'	Based on previous test suite results	Based on 'more experience'
netJ	case23	case45	case45a	case45b
netM	case26	case46	case46a	case46b

The first of the ultimate designs for each network is based on 'experience and intuition', with options chosen for what pragmatically should be the best possible set of design choices. These cases are labelled case45 and case46, for the netJ and netM designs, respectively. The second of the ultimate designs selects the options for the design based on the results of each of the previous 8 test cases. The options for the baseline cases relative to the best option cases is shown in Table 26.

Case45 differs from case45a only in that bundled rather than unit demand processing is used. Bundled demand processing ensures that demand routes are not split up across multiple routes and hence are easier to manage from an operations perspective. Case45b is essentially the same as case45a, with the exception of the use of the capture-biased demand loader rather than the balance-biased one. Case46 differs from case46a in that it uses bundled rather than unit demands, and single shortest path rather than k-way routing. The demand routes are bundled so this is the only routing option that makes sense. Case46 also uses the complex route sorter and case46a uses the simple version based on the expectation that the complex route sorter will result in higher system utilization and thus a more efficient design. Case46 uses the capture-biased demand router in order to minimize inter-ring transitions. This decision is based on the desire to minimize the amount of coupling between the ring systems. Case46b is identical to case46 except that the demand loading is balance-biased rather than capture-biased, in an effort to maximize span utilization.

As in every previous case, maximum specific progress correlates to the designs with

the lowest overall cost in both the netJ and netM test cases.

Table 26: Design Option Selection

		netJ				netM			
		Case23	Case45	Case45a	Case45b	Case26	Case46	Case46a	Case46b
Working Path Placement Considerations	Demand Bundling	Bundled	Bundled	Unit	Unit	Bundled	Bundled	Unit	Bundled
	Demand Route Distribution	Single Shortest Path	Single Shortest Path	Single Shortest Path	Single Shortest Path	Single Shortest Path	Single Shortest Path	k-way split	Single Shortest Path
	Demand Route Optimization	Hop Count Minimization	Hop Count Minimization	Hop Count Minimization	Hop Count Minimization	Hop Count Minimization	Physical Distance Minimization	Physical Distance Minimization	Physical Distance Minimization
Design Considerations	Design Technology	Single Technology	Multiple Technologies	Multiple Technologies	Multiple Technologies	Single Technology	Multiple Technologies	Multiple Technologies	Multiple Technologies
	Optimization Strategy	Direct Cost	Direct Cost	Direct Cost	Direct Cost	Direct Cost	Direct Cost	Direct Cost	Direct Cost
	Pre-Load Route Sorting	Simple	Complex	Complex	Complex	Simple	Complex	Simple	Complex
	Route Loading	Capture Biased	Balance Biased	Balance Biased	Capture Biased	Capture Biased	Capture Biased	Balance Biased	Balance Biased

In both the netJ and netM contexts, the 'Best Option' designs handily outperform their baseline counterparts. For netJ, the Best Option runs case45 and case45a better the baseline design cost by 4.9% and 7.5% respectively. Note that test case45a and case45b produce identical results, indicating that both the capture-biased and balance-biased techniques load the rings equivalently well. Comparing case45a to case45b, only demand routes 874 and 876 are loaded differently. As shown in Figure 68, case45a loads the demand route 874 span 28 segment on ring 11 and the demand route 876 span 28 segment on ring 2. In contrast, case45b loads the demand route 874 span 28 segment on ring 2 and the demand route 876 span 28 segment on ring 11. Note that ring 2 and ring 11 follow the same trajectory through the network, but ring 2 is a 2BLSR12, whereas ring 11 is a 2UPSR12. Balance-biased route loading is only applicable to BLSR rings; UPSR rings must use capture-biased loading. As a result, the minimal differences observed between case45s and case45b are as expected because only one BLSR ring is chosen in each of these designs.

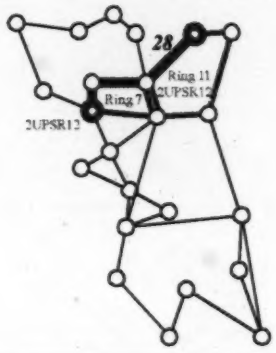
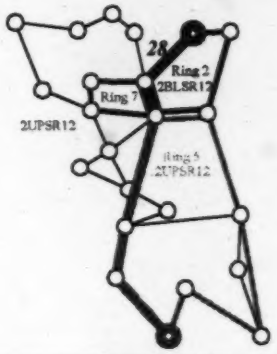
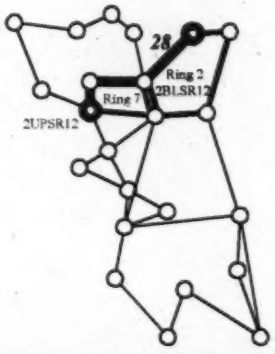
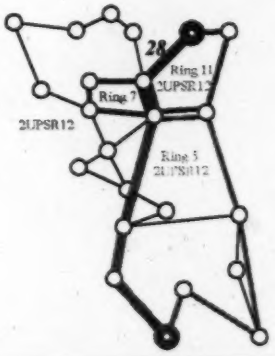
Case	Demand Route	
	874	876
45a		
45b		

Figure 68: Ring Loading Differences Between Case45a and Case45b

For netM, case46 and case46a better the baseline case by 62.2% and 60.4%. The case46 and case46b results are identical. Only the 3 BLSR rings out of the 30 rings chosen are impacted by balance-biased vs. capture-biased loading in these 2 cases, so the lack of differences between the 2 cases is not unexpected. The capture biased loader does not encounter any blocking situations that would cause segments of demands to be rejected, resulting in rejection of the entire demand.

It is interesting to note that only in netJ do the recommended design options from the previous test suites produce a better design than the 'intuition and experience' choices. In netM, the 'intuition and experience' design choices (case46 and case46b) better the pre-

vious test suite recommendations by 1.8%. This indicates that while the suggested choices from the results of the single parameter perturbation tests are good ones, they are not necessarily optimal choices.

Table 27: Test Suite 9 Results

	Network							
	netJ				netM			
	case23	case45	case45a	case45b	case26	case46	case46a	case46b
	Baseline	Best Option1	Best Option2	Best Option3	Baseline	Best Option1	Best Option2	Best Option3
Cost ($M)	5.18	4.92	4.79	4.79	411	155	163	155
Span cost ($M)	0.0920	0.0973	0.0918	0.0918	383	124	132	124
Node cost ($M)	5.08	4.82	4.70	4.70	28.2	31.2	31.2	31.2
# Rings	11	12	11	11	27	30	31	30
Avg. # hops	5.45	5.58	5.73	5.73	4.89	4.43	4.35	4.43
Avg. circumference	41.8	40.6	41.7	41.7	1181.5	688.2	707.9	688.2
# ADMs	48	56	55	55	115	97	103	97
Avg. # ADMs/ring	4.36	4.67	5.00	5.00	4.26	3.23	3.32	3.23
Transitions	1674	1386	1254	1254	1282	1636	1648	1636
Working bandwidth-distance product	21871 DS1-km	21871 DS1-km	21871 DS1-km	21871 DS1-km	779705 STS1-km	702238 STS1-km	702238 STS1-km	702238 STS1-km
Margin bandwidth-distance product	55399 DS1-km	67898 DS1-km	66292 DS1-km	66292 DS1-km	751543 STS1-km	1123128 STS1-km	730993 STS1-km	1123128 STS1-km
Protection bandwidth-distance product	77270 DS1-km	163533 DS1-km	148235 DS1-km	148235 DS1-km	1531248 STS1-km	2323344 STS1-km	2619888 STS1-km	2323344 STS1-km
Avg. specific progress	0.00324 DS1-km/$	0.00365 DS1-km/$	0.00401 DS1-km/$	0.00401 DS1-km/$	0.0009266 STS1-km/$	0.00139 STS1-km/$	0.00138 STS1-km/$	0.00139 STS1-km/$
Avg. balance	0.206	0.121	0.155	0.155	0.202	0.0976	0.110	0.0976
Avg. capture	0.926	0.927	0.922	0.922	0.921	0.930	0.930	0.930
# 2BLSR12$_{DS1target}$	11		1	1				
# 2UPSR12$_{DS1target}$		12	10	10				
# 4BLSR48$_{STS1target}$					27			
# 2UPSR48$_{STS1target}$						22	24	22
# 2BLSR192$_{STS1target}$						3	2	3
# 2UPSR192$_{STS1target}$						5	5	5

Test Suite 10: Effect of Network Topology

This test suite differs from the previously generated ones in that it is an example of a green-field application study where some practical boundary conditions are considered,

using RingBuilder™ synthesis as the evaluation tool. A collection of nodes is connected via a nearest-neighbor topology (netY) in case47, a minimally connected topology (netYm) in case47m, a sparsely connected topology (netYa) in case48, and a fully mesh-connected topology (netYb) in case49. The test suite is summarized in Table 28.

Table 28: Test Suite 10

Network	Network Type	Test Case
netYb	Fully-connected mesh	case49
netY	Nearest-neighbor connected	case47
netYa	Sparsely span connected	case48
netYm	Minimally span connected	case47m

The results are shown in Table 29. The minimally connected, Hamiltonian network topology of netYm in case47m is the least expensive design, with a cost of $1.54M. This design has the lowest number of ADMs (19) and the fewest number of ring systems (4). It has the highest balance figure of merit, the highest capture figure of merit, and the highest specific progress. The full mesh design is the second lowest cost network satisfying the demand payload, having a total cost of $1.69M. Span cost actually increases in this design relative to the other three designs, but node cost is minimized, since the inter-ring transition count drops to zero. The fully-connected mesh network also has the most installed margin bandwidth and is thus the most capable of handling increased demand. This design is made up of 5 distinct rings, the second fewest of all of the designs for this node/demand pattern. Specific progress is the highest for the fully mesh connected design.

Table 29: Test Suite 10 Results

	Network			
	netY	netYm	netYa	netYb
	case47	case47m	case48	case49
	Nearest Neighbor	Minimally Connected	Sparsely Connected	Fully Connected Mesh
Cost ($M)	2.01	1.54	2.13	1.69
Span cost ($M)	0.00909	0.0109	0.00906	0.0114
Node cost ($M)	2.00	1.53	2.13	1.67
# Rings	6	4	6	5
Avg. # hops	3.67	7.00	4.00	4.20
Avg. circumference	7.6	13.6	7.6	11.4
# ADMs	21	19	21	21
Avg. # ADMs/ring	3.50	4.75	3.50	4.20
Transitions	366	0	502	0
Working bandwidth-distance product	3511 DS1-km	5583 DS1-km	4301 DS1-km	3377 DS1-km
Margin bandwidth-distance product	4129 DS1-km	3552 DS1-km	3312 DS1-km	6223 DS1-km
Protection bandwidth-distance product	7640 DS1-km	9136 DS1-km	7613 DS1-km	9600 DS1-km
Avg. specific progress	0.00180 DS1-km/$	0.00386 DS1-km/$	0.00199 DS1-km/$	0.00202 DS1-km/$
Avg. balance	0.331	0.498	0.415	0.253
Avg. capture	0.943	1.00	0.928	1.00

Test Suite 11: Effect of Increased ADM Node Limit

This test suite explores the relaxation of the constraint on the number of active ADMs per system in order to determine whether the SONET limit of 16 active nodes is an impediment to achieving even lower cost designs. The test suite is summarized in Table 32. Case50 is a re-run of test case23 with netJ, with the ADM limit changed to 64 ADMs/ ring system. Case51, using netM, is case26 with the ADM count limit also changed to 64

ADMs/ring system.

Table 30: Test Suite 11

Network	16 ADM/ system limit	64 ADMs/ system limit
netJ	case23	case50
netM	case26	case51

The results of this test suite are summarized in Table 31. Increasing the maximum number of ADMs/system does not change the design from the baseline case. For netJ, the average number of ADMs/system remains constant at 4.36. For netM, relaxing the limit of active ADMs/system changes the design somewhat, producing 1 system with 17 ADMs, 1 more ADM than is allowed by the standard SONET network element addressing scheme. The total number of ring systems (27) remains the same as in the baseline case; the number of ADMs increases by 1 to 116. The average number of ADMs/ring increases from 4.26 to 4.30, and the margin and installed bandwidth increase somewhat. Most important is that effectively removing the ADM limit results in a more expensive design than in the ADM count limited baseline case. As always, the highest specific progress correlates with the lowest cost design. Therefore, the SONET limit of 16 active ADMs/system does not prevent the generation of efficient designs in either netJ or netM. If the demand patterns were modified, it could be possible to exploit far more than 16 active nodes on a ring.

Table 31: Test Suite 11 Results

	Network			
	netJ		netM	
	case23	case50	case26	case51
	Baseline	64 ADM	Baseline	64 ADM
Cost ($M)	5.18	5.18	411	412
Span cost ($M)	0.0920	0.0920	383	383
Node cost ($M)	5.08	5.08	28.2	28.3
# Rings	11	11	27	27
Avg. # hops	5.45	5.45	4.89	4.89
Avg. circumference	41.8	41.8	1181.5	1182.7
# ADMs	48	48	115	116
Avg. # ADMs/ring	4.36	4.36	4.26	4.30
Transitions	1674	1674	1282	1270
Working bandwidth-distance product	21871 DS1-km	21871 DS1-km	779705 STS1-km	779705 STS1-km
Margin bandwidth-distance product	55399 DS1-km	55399 DS1-km	751543 STS1-km	753079 STS1-km
Protection bandwidth-distance product	77270 DS1-km	77270 DS1-km	1531248 STS1-km	1532784 STS1-km
Avg. specific progress	0.00324 DS1-km/$	0.00324 DS1-km/$	0.000927 STS1-km/$	0.000916 STS1-km/$
Avg. balance	0.206	0.206	0.202	0.201
Avg. capture	0.926	0.926	0.921	0.922

Test Suite 12: Effect of Multiple Simultaneous Parameter Changes

This test suite explores the effect of simultaneously changing 2 parameters relative to a baseline case to determine whether there is an interaction between design parameters. The test cases comprising the test suite are shown in Table 32.

Table 32: Test Suite 12

Route Loader	Network	Pre-load Route Sorter	
		Simple	Complex
Capture Biased	netJ	case23	case33
	netM	case26	case34
Balance Biased	netJ	case41	case52
	netM	case42	case53

The results of this test suite are summarized in Table 33. For netJ, changing either the pre-load route sorting or the route loading parameter results in a lower cost design than the baseline case. Changing both of the parameters simultaneously results in a further decrease in design cost. In the case of netJ, the balance-biased route loader seems to perform better than the capture-biased route loader, even though it attempts to optimize the non-dominant span cost rather than the dominant node cost.

The most interesting result occurs for netM. In test case34, changing to complex pre-load route sorting, but keeping capture-biased route loading results in a more expensive design. This is not unexpected, because the use of capture-biased route loading in a long-haul network attempts to minimize the non-dominant node cost component at the expense of excess span cost, which masks any benefit that might come from the use of the complex route sort heuristic. The rise in cost is likely due to greedy effects of the synthesis framework.

In test case42, changing to balance-biased route loading but leaving the pre-load route sorting as in the baseline case, results in a lower cost design because the dominant cost, span cost, is being optimized. In test case53, where both the pre-load route sorter and the route loader are changed from the baseline case settings, a design is generated that is less expensive than the baseline case and is also less expensive than the result obtained in case42. This is expected, because both the pre-load route sorter and the complex route loader are optimizing the dominant cost component, span cost.

Table 33: Test Suite 12 Results

	netJ				netM			
	case23	case33	case41	case52	case26	case34	case42	case53
	Simple Route Sort /Capture Based Route Loader	Complex Route Sort /Capture Biased Route Loader	Simple Route Sort /Balance Biased Route Loader	Complex Route Sort /Balance Biased Route Loader	Simple Route Sort /Capture Biased Route Loader	Complex Route Sort /Capture Biased Route Loader	Simple Route Sort /Balance Biased Route Loader	Complex Route Sort /Balance Biased Route Loader
Cost ($M)	5.18	4.71	4.94	4.59	411	436	371	371
Span cost ($M)	0.0920	0.0719	0.0774	0.0699	383	410	346	346
Node cost ($M)	5.08	4.64	4.87	4.52	28.2	26.1	25.0	24.9
# Rings	11	8	11	9	27	29	27	27
Avg. # hops	5.45	6.13	4.91	5.22	4.89	4.86	4.67	4.67
Avg. circumference	41.8	45.0	35.2	38.8	1181.5	1179.2	1067.9	1067.9
# ADMs	48	43	44	41	115	118	106	106
Avg. # ADMs/ ring	4.36	5.38	4.00	4.56	4.26	4.07	3.93	3.93
Transitions	1674	1570	1744	1596	1282	1228	1340	1318
Working band-width-distance product	21871 DS1-km	21871 DS1-km	21871 DS1-km	21871 DS1-km	779705 STS1-km	779705 STS1-km	779705 STS1-km	779705 STS1-km
Margin band-width-distance product	55399 DS1-km	38547 DS1-km	43104 DS1-km	36862 DS1-km	751543 STS1-km	861703 STS1-km	604231 STS1-km	604231 STS1-km
Protection bandwidth-distance product	77270 DS1-km	60418 DS1-km	64975 DS1-km	58733 DS1-km	1531248 STS1-km	1641408 STS1-km	1383936 STS1-km	1383936 STS1-km
Avg. specific progress	0.00324 DS1-km/$	0.00385 DS1-km/$	0.00327 DS1-km/$	0.00392 DS1-km/$	0.000927 STS1-km/$	0.000919 STS1-km/$	0.000986 STS1-km/$	0.000987 STS1-km/$
Avg. balance	0.206	0.266	0.204	0.261	0.202	0.200	0.215	0.215
Avg. capture	0.926	0.924	0.928	0.925	0.921	0.924	0.925	0.926

Summary of Test Suite Findings

The significant results for the 12 test suites are shown in Table 34.

Table 34: Interpretation of Test Suite Summary

Test Suite	Test Parameter	General Findings/Insight(s)	Notes/Qualifiers
1	Demand Bundling/Demand Route Distribution	• Unit demand processing is more efficient than bundled demand processing • K-way routing does not generate significantly better designs than 1-way routing	• may be because of few split opportunities
2	Demand Route Optimization	• Hop count minimization routing is superior for the metropolitan network netJ • Physical distance minimization routing is superior for the long-haul network netM	• high confidence result

3	Balance-Capture Optimization	• Balance vs. capture optimization produces a slightly superior result than the baseline case for both netJ and netM. Lowest cost balance vs. capture runs had the highest specific progress
		• Balance and capture are opposing forces in the design cost
		  • optimizing for balance (fill) maximizes the transition count
		  • optimizing for capture (transitions) minimizes fill
		• Minimizing number of installed systems results in minimum cost design
		• Node cost dominates in metropolitan network; span cost dominates in long-haul network
			• takes 11 runs to find the low-cost result

4	Pre-Load Route Sorting	• Complex route sorting heuristic is superior in netJ; simple route sorting heuristic is superior in netM. Raising average fill levels in the long-haul network resulted in a higher number of deployed systems and higher cost	• consistent with "capture dominates" interpretation for metropolitan designs
5	Network Node Constraints	• Preventing transitions at a node raises overall network cost	• as expected
6	ADM Constraints	• Partial rather than full access to line bandwidth at a node raises overall design cost • Using only diversely routed point-to-point systems is extremely inefficient	
7	Route Loading	• Balance-biased route loading is superior to capture-biased route loading	• may be subject to greediness effects
8	Multiple Technologies	• The use of multiple technologies in a design resulted in lower cost designs relative to the baseline single-technology designs. In particular, the long-haul design benefited significantly from having more technology choices available to match the demand cross-sections.	• this effective strategy is ENABLED by direct cost optimization via specific progress
9	Best Option Designs	• Using the "best option' results from the previous test cases produced a good result, but not a minimum cost result. Experience and intuition guided option selection proved superior	• greediness issue and potentially parameter interaction

Table 34: Interpretation of Test Suite Summary

Test Suite	Test Parameter	General Findings/Insight(s)	Notes/Qualifiers
10	Effect of Topology	• RingBuilder™ can be used for comparative studies of network topology options	
11	Relaxation of ADM Count Restriction	• The 16 ADM per system SONET limit does not limit the efficiency of the designs produced for the metropolitan or long-haul network studied	• result highly dependent on line capacity of system.
12	Effect of Multiple Simultaneous Parameter Changes	• Greediness of technique and potential interaction of parameters lead to unexpected result	

5.3.3 Correlation Between Cost and Measured Parameters

The overall cost of a design is perhaps the single most important parameter of a survivable network design. This section examines the overall correlation of design cost to various parameters of the resultant metropolitan and long-haul networks. Cost is plotted against specific progress, protection bandwidth-distance product, installed margin bandwidth-distance product, average balance, average capture, number of ADMs, and number of rings. The test cases where the active ADM count is limited to 2 (case38 for netJ and case40a for netM) have been excluded because the 2 ADM rings are architecturally more like 1+1 DP systems than SHRs and as a result the excessive cost of the resulting designs distorts the correlation statistics significantly. Further, because of the extremely inefficient *pure capture* result produced for netM, this data point is also excluded. Pure capture optimization is unsuitable for long-haul networks because pure capture bias completely ignores system span fill, the dominant driver of system cost.

Cost vs. Specific Progress

Relative cost is plotted against specific progress for netJ and netM, in Figure 69 and Figure 70. These figures show that, in general, the higher the specific progress, the lower

the cost of the design.

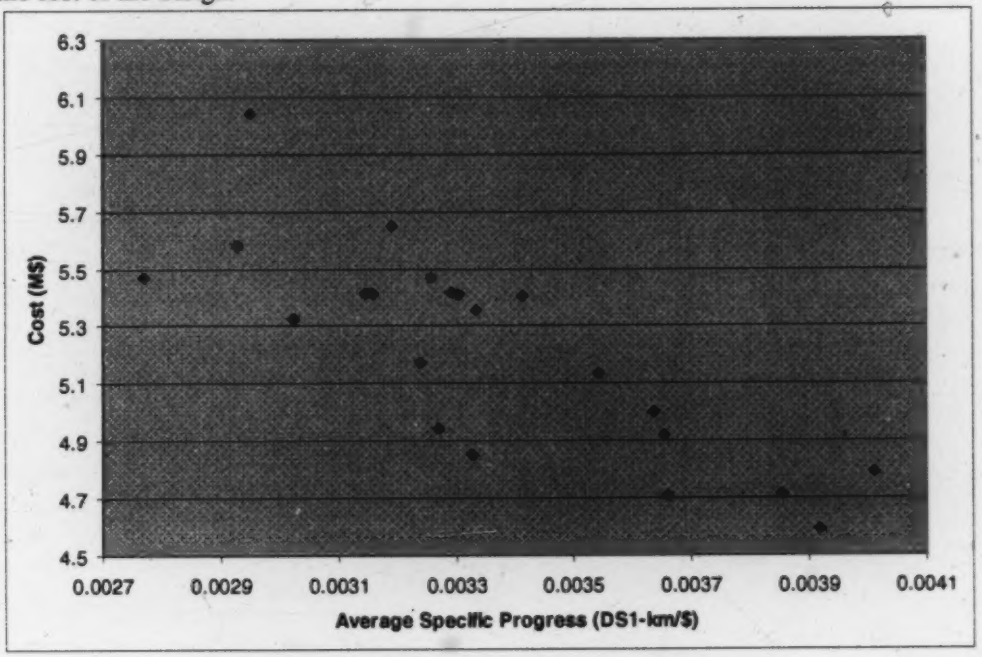


Figure 69: netJ – Cost vs. Average Specific Progress

The correlation coefficient, r, for cost vs. average specific progress for the netJ runs is -0.814, indicating that the specific progress is inversely correlated to cost. The correlation coefficient for the equivalent netM runs is -0.846. This finding and the low scatter within the main clusters above strongly confirm the basic idea and hypothesis of synthesizing ring networks iteratively on the basis of direct cost/specific progress. Separating the span and node costs for netJ as shown in Figure 71 reveals that average specific progress correlates well (r = -0.836) to the node cost, the dominant cost component for netJ. The correlation of average specific progress to the span cost is -0.264. The opposite situation occurs for netM as shown in Figure 72, with average specific progress correlating well to the span cost (r = -0.653) and not at all to the non-dominant cost component, node cost (r = 0.0219).

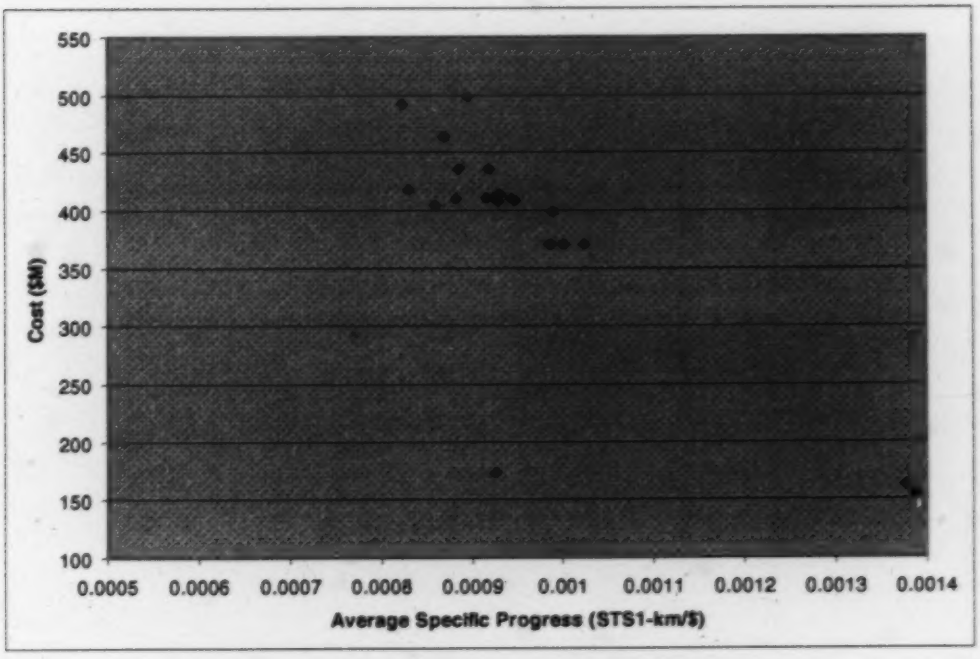


Figure 70: netM -- Cost vs. Average Specific Progress

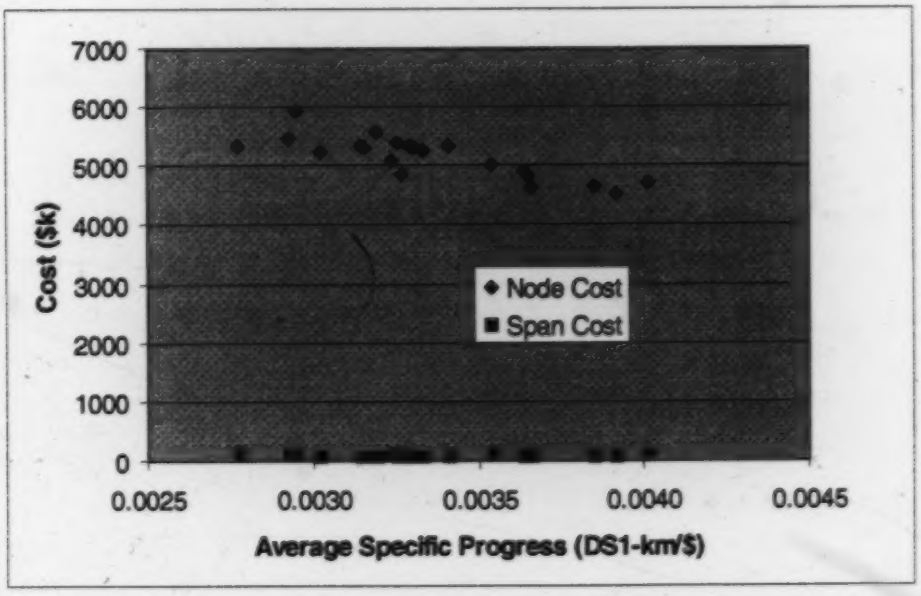


Figure 71: netJ -- Node and Span Cost vs. Specific Progress

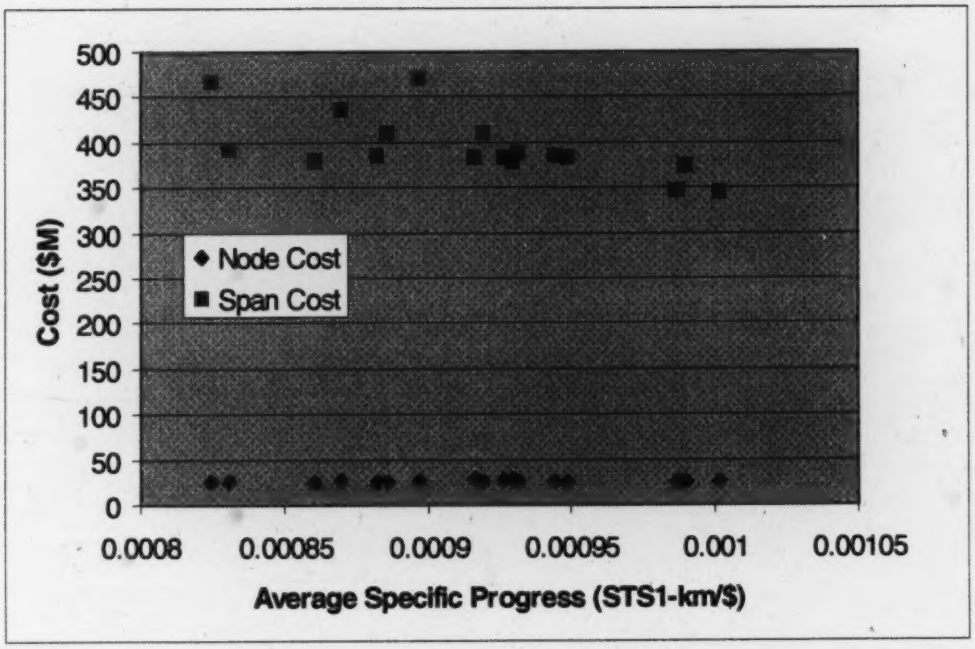


Figure 72: netM -- Node and Span Cost vs. Average Specific Progress

Cost vs. Protection Bandwidth-Distance Product

Figure 73 shows cost as a function of protection bandwidth-distance product for netJ and reveals that little correlation exists between cost and installed bandwidth for this network.The correlation coefficient for cost vs. protection bandwidth distance product for netJ is -0.260. Figure 74 shows equivalent relationship for netM. The correlation coefficient in this case is -0.759. The correlation coefficient becomes 0.999 if the 4 low-cost designs represented by the data points in the lower right-hand corner of the graph are ignored. For netM, this indicates that cost is directly correlated to the amount of installed bandwidth. For this network, the lowest cost designs are represented by the data points in the lower right-hand corner of Figure 74. These data points are for the multi-technology runs done for netM, clearly showing the benefits of a multiple-technology design strategy.

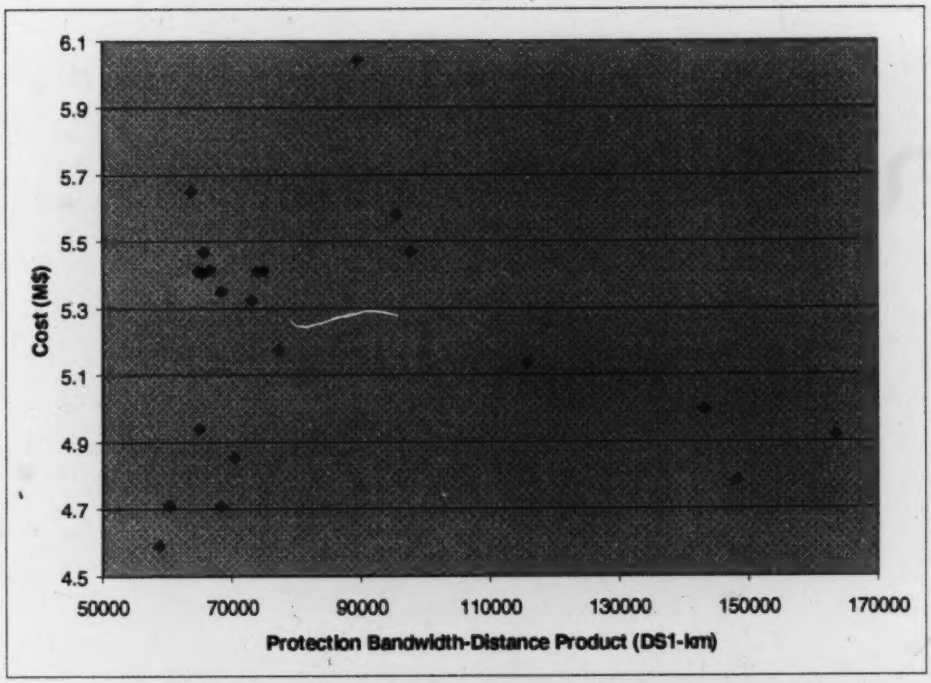


Figure 73: netJ -- Cost vs. Protection Bandwidth-Distance Product

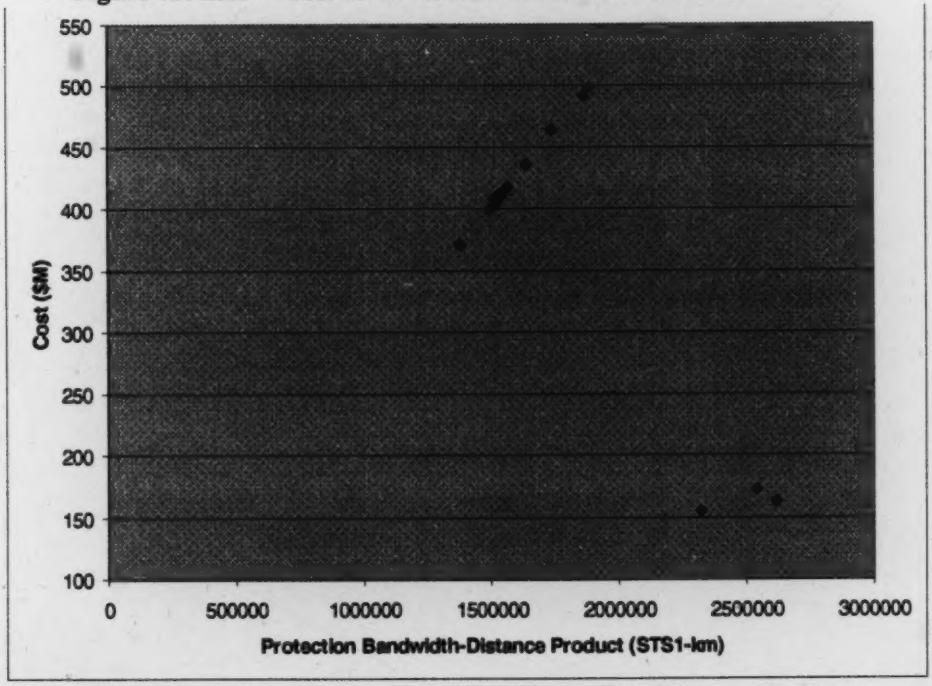


Figure 74: netM -- Cost vs. Protection Bandwidth-Distance Product

Cost vs. Installed Margin

No distinct correlation is observed between the installed margin and the resulting overall system cost for netJ as shown in Figure 75. The correlation coefficient of cost vs. margin bandwidth-distance product for netJ is 0.114. Figure 76. shows the equivalent plot for netM. In this case, the correlation coefficient is -0.0217, respectively. If the four low-cost data points (from the multi-technology runs for netM) in the lower right hand corner are ignored, the correlation coefficient becomes 0.994, indicating a strong correlation between installed margin and cost.

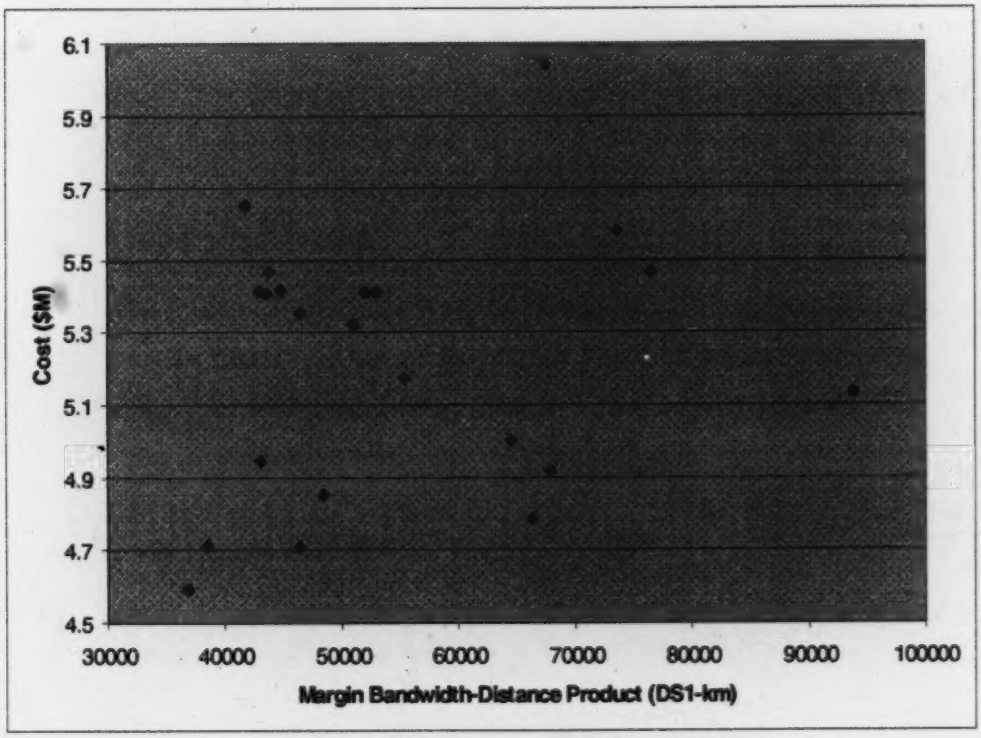


Figure 75: netJ – Cost vs. Installed Margin Bandwidth-Distance Product

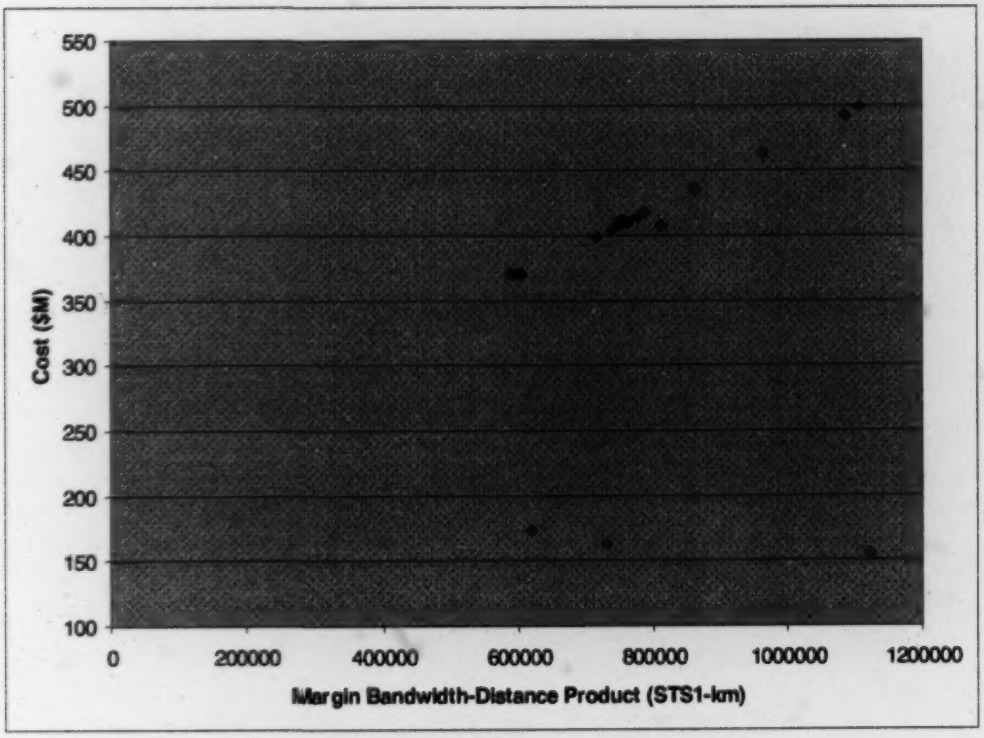


Figure 76: netM – Cost vs. Installed Margin Bandwidth-Distance Product

Cost vs. Average Balance

Little correlation is observed between overall system cost and average balance for the designs generated for netJ as shown in Figure 77. The correlation coefficient for cost vs. average balance are -0.139 for netJ. For netM, as the correlation coefficient for the designs plotted in Figure 78 is 0.789. Removing the multi-technology designs from consideration changes this value to -0.685. This shows the importance of balance optimization for long-haul networks.

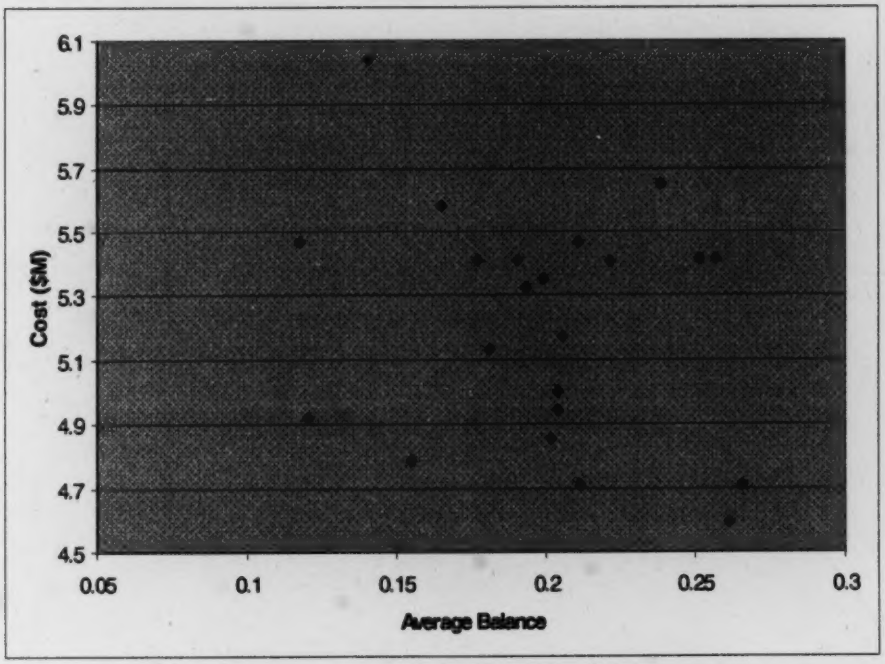


Figure 77: netJ – Cost vs. Average Balance

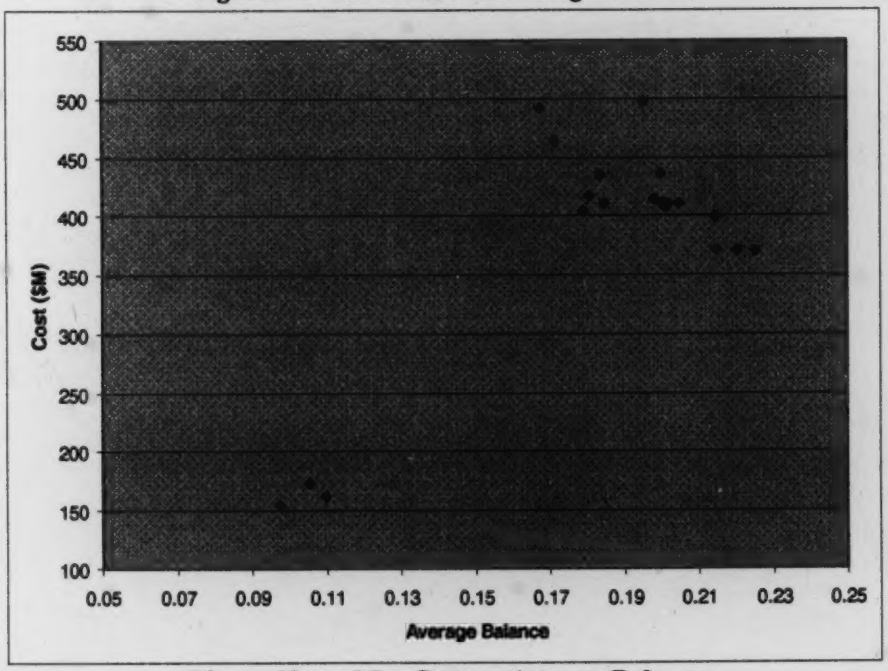


Figure 78: netM – Cost vs. Average Balance

Cost vs. Average Capture

The correlation of total system cost to average capture is examined in Figure 79 and Figure 80. Little correlation is observed between cost and the average capture figure of merit for either netJ or netM, confirmed by the calculated correlation coefficients of -0.00558 and -0.0577. Removing the multiple technology designs from consideration changes the correlation coefficient for netM to -0.0289.

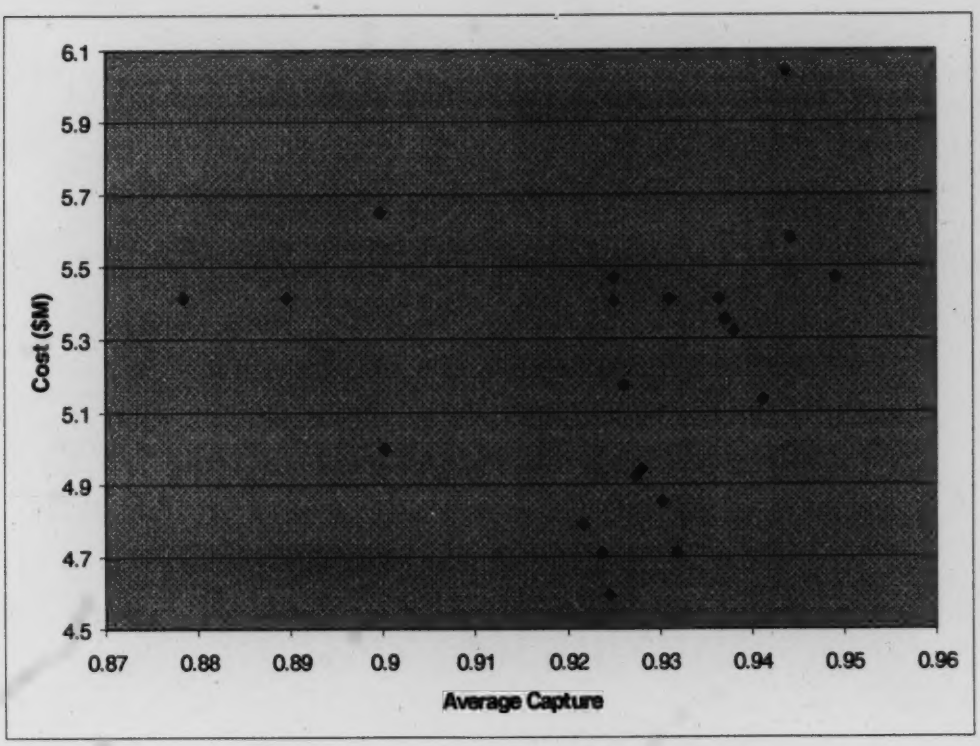


Figure 79: netJ -- Cost vs. Average Capture

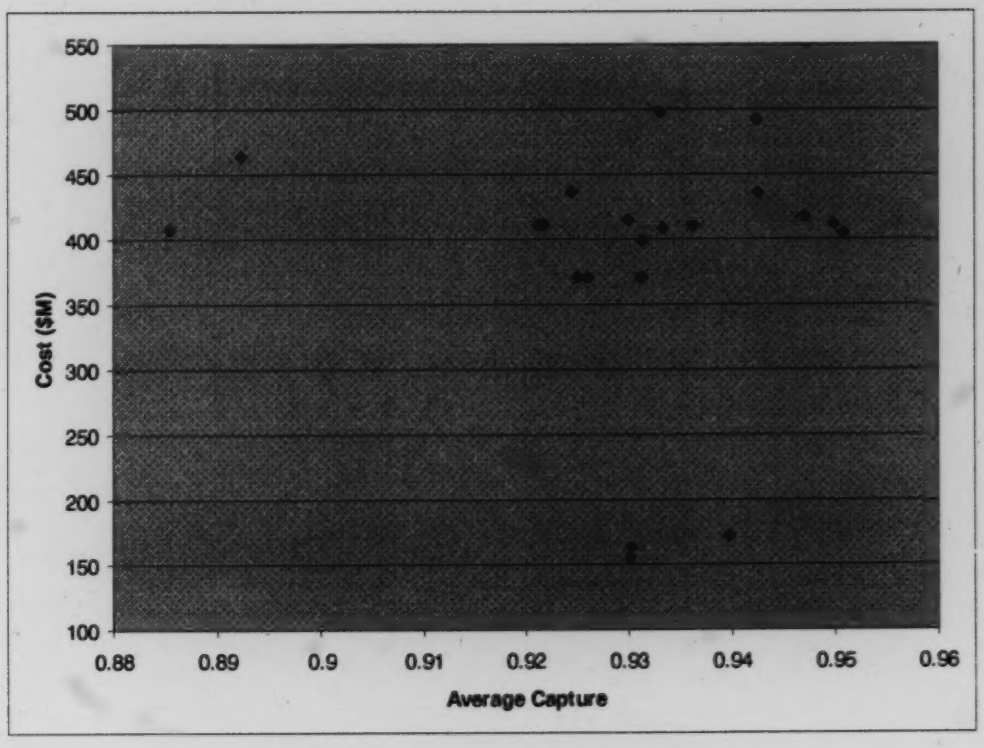


Figure 80: netM -- Cost vs. Average Capture

The cost vs. average balance and cost vs. average capture results just presented are somewhat unintuitive because it has been demonstrated that balance and capture are good optimization metrics for ring synthesis [10]. The impression is caused by be the skewing of the *average* values for the design due to the final rings chosen in the design process. To illustrate this, Figure 81 and Figure 82 show the balance, capture and specific progress for each of the rings chosen for the baseline designs for netJ and netM. The final rings selected have low balance and specific progress but have high capture. This means that a ring can be extremely poorly utilized (low balance and specific progress) but the remaining segments of a demand route are all carried (high capture). Thus, average capture for an entire design will be increased as the last rings are chosen. Similarly, the average balance and specific progress figures will be reduced. When balance and capture are used as optimization metrics, skewing does not occur because the balance and capture are calcu-

lated for individual ring candidates, and *not* averaged across an entire set of rings.

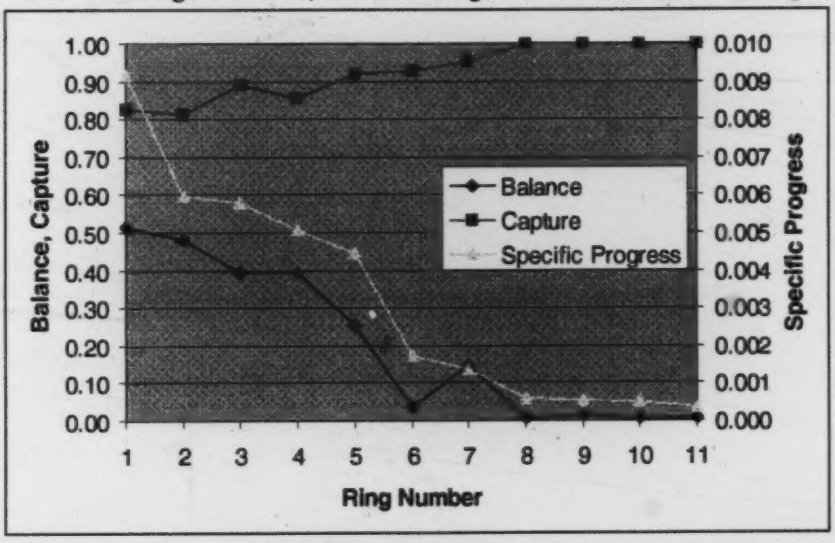


Figure 81: netJ -- Balance, Capture, and Specific Progress vs. Ring Number for Case23 Design

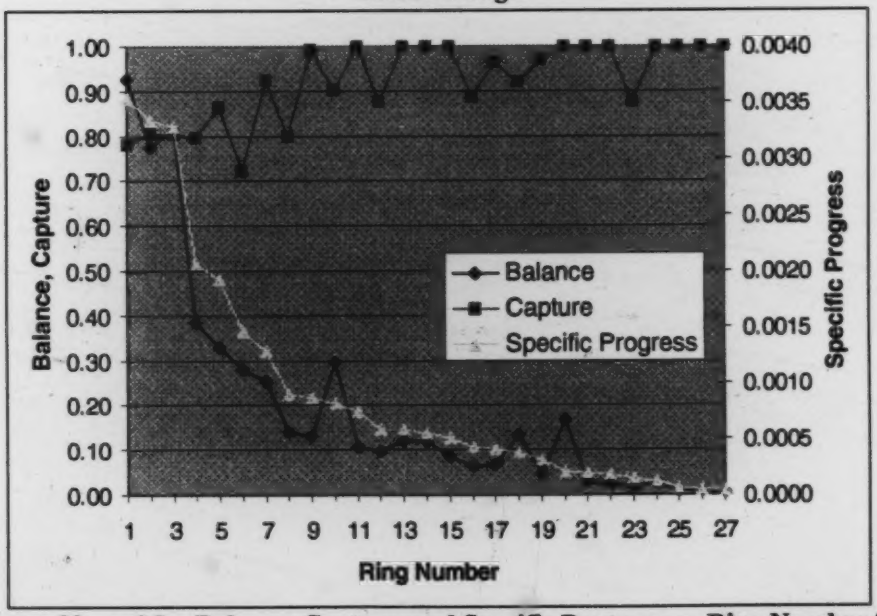


Figure 82: netM -- Balance, Capture, and Specific Progress vs. Ring Number for Case26 Design

Figure 83 and Figure 84 show the cost, progress and specific progress vs. ring number for the two baseline designs. In both cases, 90% of the total working bandwidth distance product for the design was accommodated by the first subset of the rings chosen.

The latter stages of the designs are comprised of rings having low specific progress and thus are inefficient. This highlights the limitations of the greedy technique used; early ring choices are extremely efficient, but these ring choices result in ring choices later in the design based on a much smaller and more limited set of alternatives. Techniques to overcome this aspect of the greedy synthesis technique are required to further optimize the designs generated.

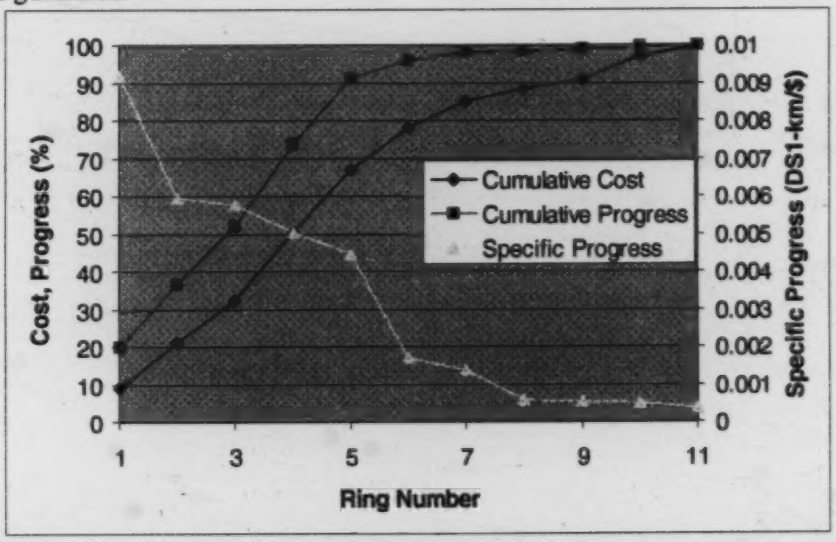


Figure 83: netJ -- Cost, Progress, and Specific Progress vs. Ring Number for Case23

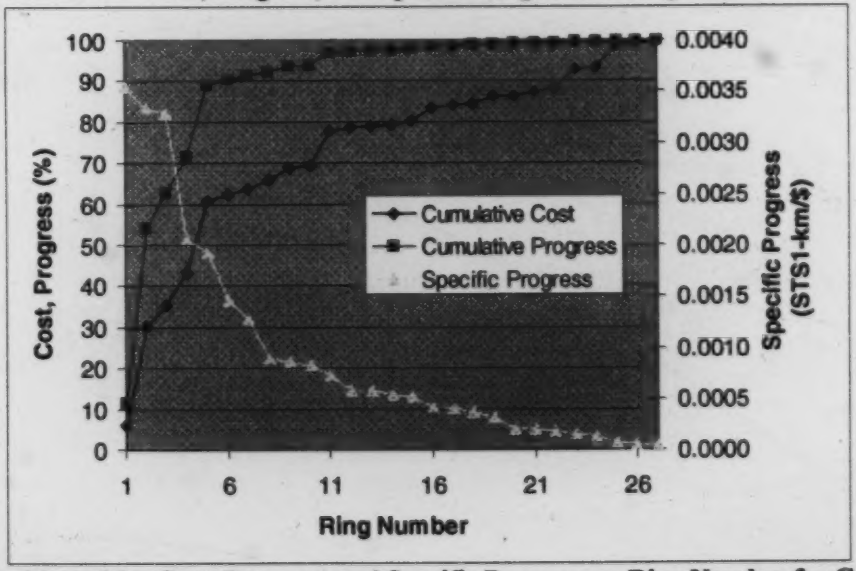


Figure 84: netM -- Cost, Progress, and Specific Progress vs. Ring Number for Case26

Cost vs. Number of ADMs

Figure 85 (netJ) and Figure 86 (netM) show a correlation between total system cost and the number of installed ADMs for both netJ and netM, with calculated correlation coefficients of 0.534 and 0.752. The latter figure improves to 0.831 if the multi-technology designs are removed from consideration.

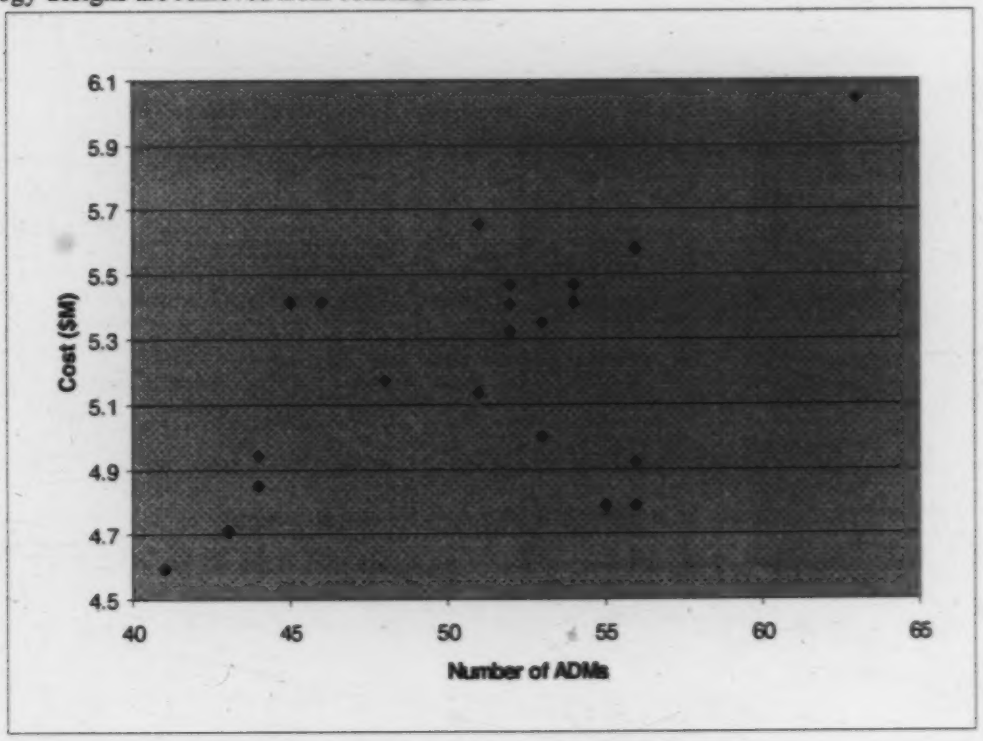


Figure 85: netJ -- Cost vs. Number of ADMs

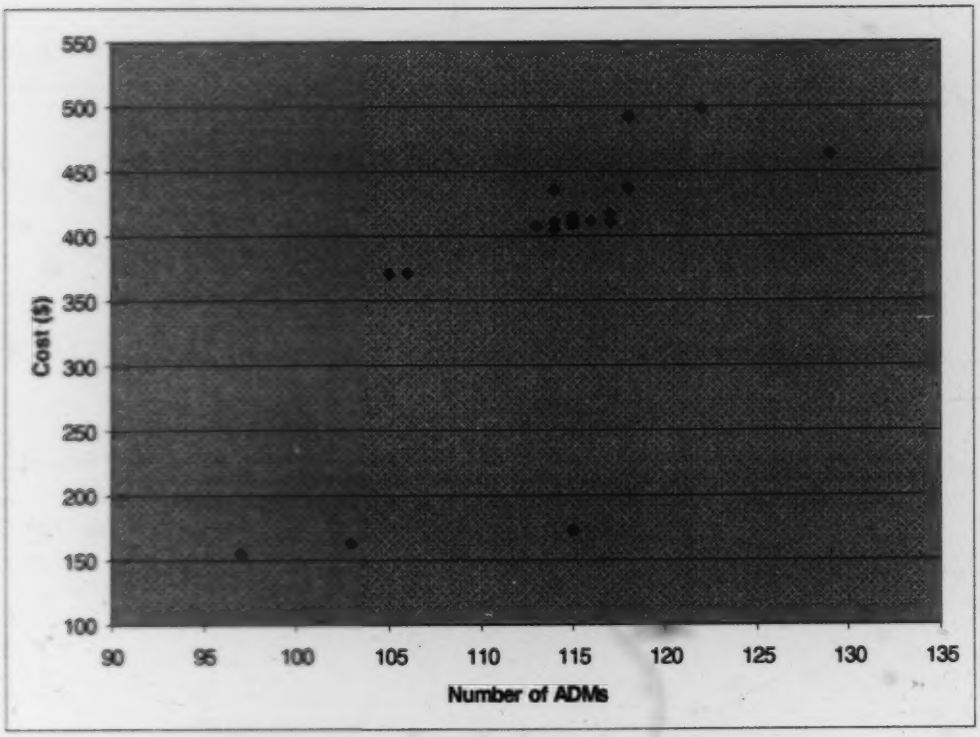


Figure 86: netM -- Cost vs. Number of ADMs

Cost vs. Number of Rings

The correlation between total system cost and the number of ring systems deployed is examined in Figure 87 and Figure 88. Some correlation between cost and the number of ring systems is identified for netJ, as the correlation coefficient is 0.542. For netM, the correlation coefficient is -0.566, changing to 0.409 when the multi-technology designs are removed from consideration.

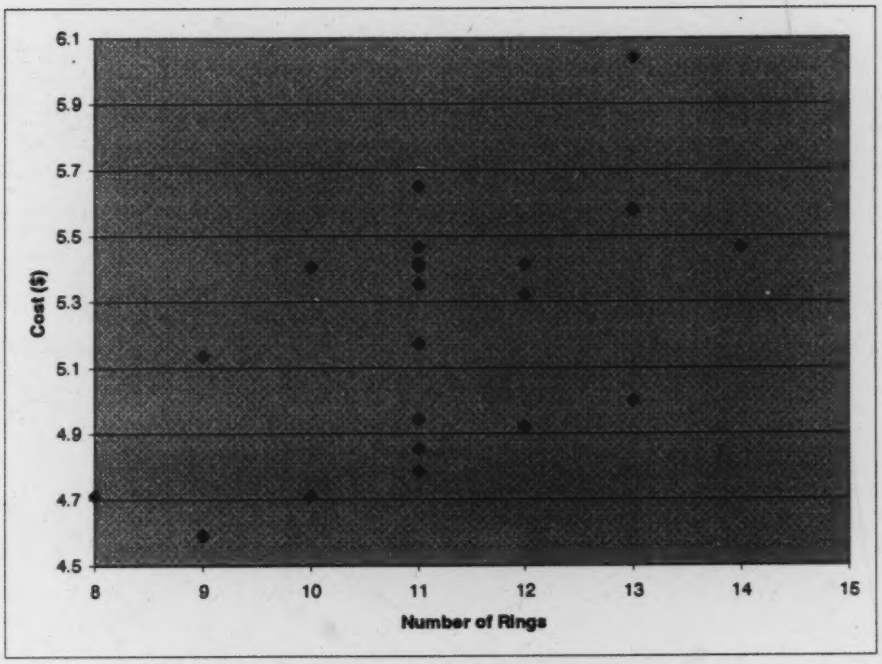


Figure 87: netJ -- Cost vs. Number of Rings

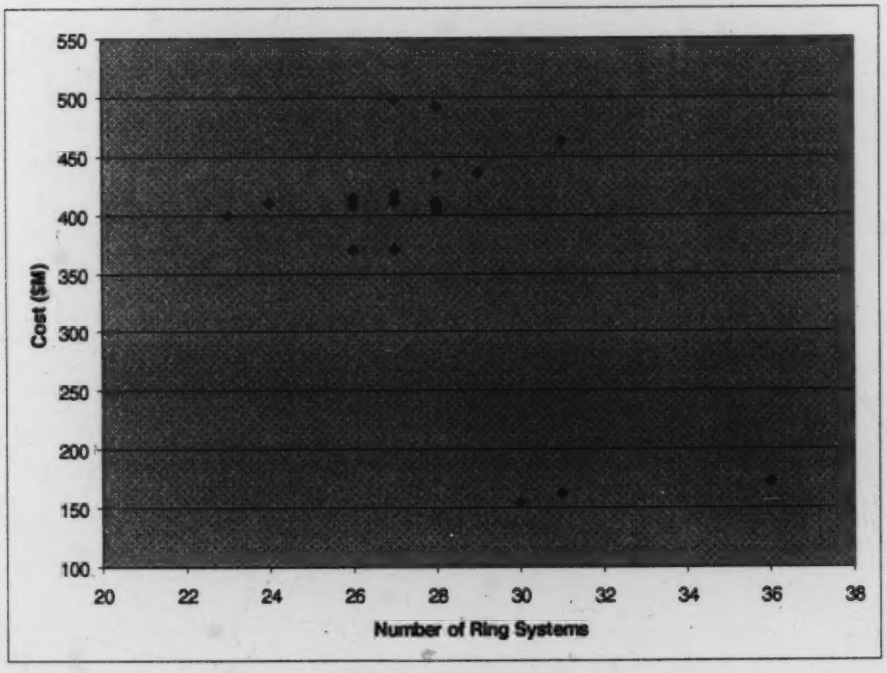


Figure 88: netM -- Cost vs. Number of Rings

5.4 Conclusions

In this chapter, the performance of the RingBuilder™ framework was evaluated using both metropolitan and long-haul network test cases. Both balance vs. capture optimization and specific progress optimization were characterized. Specific progress optimization was found to be the marker of any superior design result regardless of how the design was obtained. It was also illustrated how RingBuilder™ runs can be used to study topology design issues. RingBuilder™ generated low-cost designs for each candidate presented to it.

Several design principles were discovered. Unit demand processing results in more efficient designs. Demand splitting through the use of k-way routing did not prove to provide much benefit over simple single shortest path routing with the networks studied. Note, however, that in the long-haul network, the k-way routing strategy may have performed better if it was doing splitting of equal physical distance routes. Hop count minimization routing was the preferred technique for metropolitan networks, whereas physical distance minimization was shown to be superior in the long-haul network context. In both the metropolitan and long-haul studies, the lowest cost designs did not have the minimum number of ring systems. Using complex route sorting prior to route loading did not always result in lower cost designs. Limiting the amount of add/drop bandwidth at any one node resulted in higher cost metropolitan and long-haul designs. Using solely 1:1 diversely protected systems is inefficient in both the metropolitan and long-haul contexts. Balance-biased demand loading is superior to capture-biased demand loading. Design option selections that are good in isolation, may not be optimal when considered with other options. This may be due to parameter interaction but is most likely a greediness effect.

Several properties of metropolitan and long-haul networks were discovered. Node costs dominate over span costs in metropolitan networks, so it is prudent to minimize the utilization of terminal equipment by maximizing capture. Span costs dominate over node costs in long-haul networks, so design cost can be minimized by minimizing installed bandwidth-distance products. Maximizing balance maximizes system utilization and therefore minimizes installed bandwidth. The best metropolitan designs had high cap-

ture, but not minimum installed bandwidth. The best long-haul designs had minimum installed bandwidth. The SONET limit of 16 active nodes per ring is not a barrier to efficient network design.

Average specific progress is a suitable metric with which to rank competing multi-technology designs. In all cases examined in this chapter, the lowest cost design had the highest average specific progress. Specific progress optimization is found to be superior to balance vs. capture optimization in both the metropolitan and the long-haul network study cases. Multiple technology designs, enabled by the development of the direct cost optimization via specific progress, were superior to the single technology designs from either of the optimization strategies.

6. Concluding Discussion

A detailed exploration of the synthesis of ring-based survivable networks has resulted in the development of a multi-technology ring synthesis framework called RingBuilder™. This framework addresses many of the network, system and design issues of self-healing ring network design.

6.1 Overall results

Due to the broad scope of this work, observations can be made about three distinct areas: the software development, the characteristics discovered about two classes of telecommunications networks, and the properties discovered about the RingBuilder™ synthesis framework.

6.1.1 Software Development Observations

The software design and implementation for RingBuilder™ took approximately 36 person-months of effort. Because of the scope of the software development effort, several key characteristics about the development of such a large system were learned. Key learnings in the areas of the development environment, the modular development approach, software testing, and manual verification activities are described below.

RingBuilder™ was written entirely in 'C', using GNU development tools. The software was debugged and tested on three different computing platforms. All of the early development and the memory leak testing work was performed on Sun™ SPARC™ running Solaris™ 4.1.3[1]. Later, development was continued using DEC™ Alpha™ workstations under OSF1[2]. The last development work, all functionality testing, and all experiments were done on an IBM 300G Pentium II 266 workstation running Red Hat™ Linux™ 4.2[3]. With the exception of the slightly different implementations of the quicksort routines on each of the platforms, and the lack of memory leak testing tools on all but the

1. Sun and Solaris are trademarks of Sun Microsystems and SPARC is a trademark of SPARC International
2. DEC and Alpha are trademarks of Digital Equipment Corporation
3. Red Hat is a trademark of Red Hat Software, Inc. Linux is a trademark of Linus Torvalds

Sun™ workstations, the actual choice of platform was transparent to the developer. Xemacs release 19 was available on all of these environments the editing/debugging tool.

Although more modern languages (C++, Java) could have been used, 'C' was chosen as the programming language for this project. It was the 'language available by default', but it also incorporates required programming features. The ability to dynamically allocate and deallocate memory was a key requirement to implement the RingBuilder™ algorithm, due to the relatively large amounts of storage required per iteration and per design in the greedy iterative framework. The manual memory allocation and deallocation used in C proved to be relatively easy to manage, with the exception of the syntactical difficulties with pointer dereferencing that arose time and again. The techniques involved in passing parameters between functions evolved as the software development process proceeded. Indirect referencing of entities became the standard way for functions to interact. Despite some arcane syntactical constructs, this proved to be a very efficient way of passing information between functional blocks.

The architecture of RingBuilder™ was synthesized using a top-down hierarchical approach using a **modular development strategy**. This proved very successful, as modules could be coded and tested without the presence of other related functional blocks. Strictly adhering to rules of thumb about limiting the number of layers of nested branches to 4 or less, and to limiting the length of each function to <100 lines of executable code, resulted in an efficient, maintainable set of functional blocks.

The information interfaces between the various functions evolved as code development progressed. References to data structures were passed rather than long, hard to maintain lists of individual variables. Files were used for input/output operations, and all I/O was left to higher level functions. For instance, the RingBuilder™ main routine was responsible for file opening and closing, calling the ring design function (chooseRings) and little else. Such functional partitioning made fault isolation straightforward.

The **software verification process** ensured that individual modules were functionally verified as they were added to the RingBuilder™ framework. Memory leak testing was performed once all of the main modules were integrated. Fault isolation was straightforward, making this a very effective way to develop and integrate the code.

The **manual verification** of the RingBuilder™ framework was a critical, yet labour-intensive step in the development activity. The initial intent was to provide a set of manual baseline cases to which RingBuilder™ results could be compared, in order to verify the synthesized results. Twenty-two test cases were designed, all based on a network small enough to permit manual designs to be generated. These test cases proved invaluable for software bug isolation. Not all software bugs were caught via the 22 test cases, however. Several subtle problems remained until larger network experiments were started.

6.1.2 General Findings About Ring Network Designs

Several fundamental characteristics of metropolitan and long-haul networks were observed through the course of this work. The following paragraphs detail the key findings with respect to balance-capture optimization, the dominant costs of the metropolitan and long-haul networks, and the impact the SONET standard limit of 16 active ADMs per ring has on the designs.

Balance and capture were shown to be opposing forces in the ring network design problem [9]. Optimizing for balance via system utilization maximization tended to produce excess inter-ring system transitioning. Optimizing for capture minimized the inter-ring transiting, but at the expense of lower overall system utilization.

The **node equipment costs** were determined to be the dominant costs in the metropolitan network studied. As a result, design options that minimized the amount of terminal equipment required resulted in minimum cost designs.

Span costs were determined to be the dominant costs in the long-haul network studied. Design option selections that minimized the amount of fibre utilization through minimization of total installed bandwidth-distance resulted in minimum cost long-haul network designs.

Hop count minimization during demand routing is key to minimizing the amount of terminal equipment and hence minimizing the design costs for metropolitan networks. Physical distance minimization is key to minimizing the amount of installed capacity present, a priority of particular importance in the design of minimum cost long-haul networks.

The relaxation of the SONET standard maximum of **16 active ADM nodes per ring**

system revealed that there was little impact on the overall design when the ADM count limit was removed. Most ring systems in the designs generated had far fewer than 16 ADMs. This result is highly dependent on the demand pattern, however, and it is possible to encounter cases where rings with far more than 16 active nodes could be extremely efficient.

6.1.3 Design Synthesis Observations

Several observations can be made with respect to the use of the RingBuilder™ framework to generate 'real-world' ring network designs.

Specific cost optimization is superior to balance-capture optimization in both single technology and multiple technology designs. The designs generated with specific cost optimization are as efficient or more efficient than those generated through balance-capture optimization. Further, specific cost optimization is substantially quicker than balance-capture optimization because a minimum cost design is produced in a single run rather than requiring a sweep of runs as is the case with balance-capture optimization.

Multiple technology optimization is superior to single technology optimization because it allows system sizes to be more closely matched to the demand payloads. This results in higher overall system utilization and ultimately fewer total systems, both of which are significant contributors to minimum cost designs.

In the metropolitan and long-haul networks examined in this work, it was found that single shortest path routing was as suitable as k shortest path routing in generating ring designs. This is not a general result, however, because the demand splitting process is highly dependent on the demand pattern. If demand bundling was specified as an option, k shortest path routing could not be used.

Generating designs using **unit demands** sized at the demand management quantity level results in more efficient utilization of the system installed capacity. However, sometimes bundled demands have to be used, and an overall efficiency penalty is paid.

Balance-biased loading was more suitable than capture-biased loading in both the metropolitan and long-haul network studies carried out in this work. The diminished fill resulting from the use of the capture-biased route loader was not offset by corresponding inter-ring transition cost reductions, as expected. This result is demand pattern depen-

dent.

In the heuristic-based route loaders used in RingBuilder™, it was found that generally the use of the complex route loader resulted in designs that were superior to those generated with the simple route loader.

Experiments were performed on each of the possible design parameters in isolation. An inferior long-haul network design resulted when all of the 'best' design parameter decisions were taken together in a RingBuilder™ design. This is most likely an artifact of the greediness inherent in this technique, but it may point to some design parameter interaction.

This work showed how RingBuilder™ can be used to **analyze network topology options**. RingBuilder™ was used to generate minimum-cost designs for several different topology scenarios. Such an activity can be useful to a planner if central office interconnecting structure plans are being evaluated.

6.2 Uses and Adaptations for Present Work

The RingBuilder™ software suite as presented here can be used by experienced telecommunications planners to design ring based transport networks. The combination of the intuition and experience of the planner combined with RingBuilder™'s ability to generate multiple network designs quickly can result in significantly improved network designs.

6.3 Future Research

The ring network design problem, as well as telecommunications transport network design methods in general, remains a rich area for future research. Even fractional improvements in network design costs can have significant positive impacts for telecommunications carriers. The main extensions of the existing work include transition design, mixed mode design, and multi-wavelength (lambda) optical network design.

6.3.1 Transition Design Capability

Enhancing the RingBuilder™ synthesis framework with the capability to grow ring network designs from existing network designs, or to migrate to ring network topologies

from mixed linear/ring network designs would be a very valuable additional set of capabilities. Network architects rarely have the luxury of starting with 'green field' design scenarios; invariably, they have to cost-effectively evolve existing networks into target designs.

The ability to take an existing ring network design, and use its margin capacity to satisfy a set of incremental payload demands would be a very useful extension of the existing framework. Transition design, as this is called, has more utility for telecommunications network planners than does 'green field' design because it is more likely for a planner to evolve an existing network or subnetwork than to create an entirely new entity.

Similar to migrating from an all ring design, the ability to cost-effectively transition from a mixed ring/linear network to a target network design would also be a very useful capability. Existing telecommunications networks consist of transport equipment configured in linear, mesh, and ring configurations. The ability to take existing systems and redeploy them or augment them is key to the continuing efficient operation of the telecommunications infrastructure.

6.3.2 Improved Cost Models

The span cost model could be improved with the addition of a repeater costing module that would calculate the positioning and required quantity of inter-ADM repeaters on the fully-loaded ring systems. This would be an improvement over the current approach where all span costs are represented in a linear, distance-based fibre cost.

6.3.3 Mixed Mode Design

Extending the greedy iterative framework beyond ring-only synthesis to a generalized synthesis system in which rings, linear systems and mesh constructs are all considered would be a research problem with significant scope and complexity. The representation of linear systems and mesh elements in an equivalent manner to the way rings are presented in RingBuilder™ is a significant issue that must be addressed at the outset.

6.3.4 WDM Network Design

The logical partitioning of optical pipes through the use of multiple wavelengths of light is becoming a very important issue as telecommunications providers exhaust their

fibre plant due to rapidly increasing communications traffic. The extension of the RingBuilder™ framework to logically exploit the use of multi-wavelength network elements would be a research problem that could have immediate positive impact on the design of optical networks for telecommunications providers. The extension of the framework goes beyond the logical partitioning of the network spans; network elements are evolving and are soon expected to be able to interchange wavelengths as easily as they interchange timeslots today.

6.4 Conclusion

The dependence of society on the telecommunications infrastructure is increasing. The requirement that this infrastructure be survivable and efficiently implemented has led to the use of self-healing ring networks. The design of these networks is extremely complex, despite the relative simplicity of the individual rings. To date, most of the approaches to the problem have been based on analysis, rather than synthesis, and so the designs generated depend heavily on the undocumented experience and intuition of the planner. Existing automated synthesis techniques do not address all of the required aspects of the design problem. RingBuilder™ efficiently generates optimized designs by simultaneously considering network, system, and design issues. The framework is modular, extensible, and is readily usable to study further aspects of the design problem. Most importantly, it is capable of generating real, cost optimized network designs for real networks encountered by planners today.

References

[1] T. Flanagan, "Fiber Network Survivability," IEEE Communications Magazine., Vol. 28, No. 6, Jun. 1990, pp.46-53.

[2] GR-1230-Core, *SONET Bidirectional Line-Switched Ring Equipment Generic Criteria*, Bellcore, Issue 2, November 1995.

[3] P. Mateti, N. Deo, "On algorithms for enumerating all circuits of a graph," SIAM J. Comput., Vol. 5, No. 1, Mar. 1976.

[4] G.D. Morley, W.D. Grover, "A Comparative Survey of Methods for Automated Design of Ring-based Transport Networks," TRLabs Technical Report TR-97-04, Issue 1.0, Jan. 1998.

[5] GR-1400-Core, *SONET Dual-Fed Unidirectional Path Switched Ring (UPSR) Equipment Generic Criteria*, Bellcore, Issue 1, Revision 1, October 1995.

[6] R. Iraschko, Internal TRLabs Networks Systems drawing, 1996.

[7] Darryl Grainger, Doug McKeen, Tien Shiah, "Introductory Technical Guideline to Sonet Rings," AGT Limited Internal Document, 17 May 1992.

[8] Tektronix Inc., *SONET Telecommunications Standard Primer*, Internet document http://www.tek.com/Measurement/App_Notes/SONET/, March 1997.

[9] J. B. Slevinsky, W.D. Grover, M.H. MacGregor, "An Algorithm for Survivable Network Design Employing Multiple Self-healing Rings," *Proc. IEEE Globecom '93*, Dec. 1993, pp. 1568-1573.

[10] W.D. Grover, J. B. Slevinsky, M.H. MacGregor, "Optimized design of ring-based survivable networks," *Can J. elect. & Comp. Eng., Vol.20 No. 3, 1995*, pp. 139-149.

[11] Northern Telecom, Transition Planner User's Guide, 1991.

[12] J. Heitkotter, editor, "The Hitch-Hiker's Guide to Evolutionary Computation," comp.ai.genetic FAQ, Issue 1.07, September 1993.

[13] W.Misskey, "Simulated Evolution and its Application to Multiple Self-healing Ring Network Design," Presentation to TRLabs, 17 February 1994.

[14] W. Misskey, "Application of Genetic Algorithms to Multiple Self-healing Ring Network Design - Chromosome Structure," Presentation to TRLabs, 06 April 1994.

[15] T.-H. Wu, Fiber Network Service Survivability, Artech House Inc., 1992, pp. 308-318.

[16] O. J. Wasem, "Optimal topologies for survivable fiber optic networks using SONET self-healing rings," Proc. IEEE Globecom '91, Dec. 1991, pp. 2032-2038.

[17] Jianxu Shi, John P. Fonseka, "Design of Hierarchical Self-healing Ring Networks," Proc. IEEE ICC'94, 1994, pp.478-482.

[18] L.M. Gardner et al, "Techniques for Finding Ring Covers in Survivable Networks," Proc. IEEE Globecom'94, 1994, pp.1862-1866.

[19] J.B Slevinsky, "RingBuilder Version 2 Data Structure Organization and Module Implementation," TRLabs Technical Report TR-98-02, 1998.

A. RingBuilder™ Implementation Testing

RingBuilder™ is a complex, multiple module software system that processes extremely large quantities of data. It was essential to rigorously test RingBuilder™, so that planners can have confidence in the designs it generates. This appendix describes the testing methodology and the test suites used to verify the operation of the software.

A.1 Test Aim

RingBuilder™ designs are so intricate that only a cursory analysis of the design output can be performed. The intent of the software testing process is to gain a high level of confidence that the designs produced are technically feasible and complete. This is a multi-stage process where the individual modules of the main subsystems of RingBuilder™ are examined, the subsystems are validated, and the design output of RingBuilder™ as a whole is verified. The software testing process confirms that RingBuilder™ efficiently utilizes and releases computing resources, showing that it can accommodate the very large memory requirements of real-world designs.

A.2 Testing Methodology

The testing methodology is structured to first test individual modules and subsystems, then memory leaks, and finally, the entire RingBuilder™ program using a network that is small enough so that manual verification is feasible.

A.2.1 Module Testing

Each module of RingBuilder™ was tested for functionality as it was built. It was not necessary to write special programs to test these modules in isolation. Because of the top-down design of RingBuilder™, the program itself could be used. Thus, the first module tested was the RingBuilder™ main program. It opens files, processes control information, and then calls the cycle finding and ring selection operations which contain the core functions of RingBuilder™.

The module testing process cocentrated on the verification of inter-module interfaces and the operation of the logic vthin the module. Input and output was examined to ensure that desired information was eing processed and created. Diagnostic print statements, added to the modules for testig, remain in the code and can be enabled during the compilation process for any of the mjor diagnostic functions. This set of optional functions facilitates code debugging whe anomalies are identified.

Table 35: Conditiont Compilation Diagnostic Options

Conditional Compilation Class	Conditional Compilatin Option	Purpose
Memory Utilization Diagnostics	mallocDebug	Verification of memory allocation processes
	leakTest	Check for improper/missing memory de-allocation
RingBuilder™ Functional Diagnostics	cycleDiag	Examination of Cycle set constituents
	cycleLoaderBLSRDia	BLSR Cycle Loader diagnostics
	cycleLoaderBLSRCapureDiag	BLSR Capture Cycle Loader diagnostics
	cycleSpecificRouteDig	Cycle subset generation diagnostics
	everyCycleEtaDiag	Per-cycle per-iteration cycle figure of merit diagnostics
	ForwardSpanNodeChckDiag	Cycle Loader sub-module test diagnostics
	iterationTerminalStats	RingBuilder™ iteration level diagnostics
	optimizationStrategyDag	Optimization strategy selection diagnostics
	routeSortDiag	Cycle specific route sorting diagnostics
	spanSpecificRouteDig	Span based route list generation diagnostics
	TerminalNodeCheckDag	Cycle Loader sub-module test diagnostics

A.2.2 Memory Leak Testing

Memory leak testing was perfomed after all RingBuilder™ modules had been created and tested. Sun™ SPARC™ Solris™ 4.1.3 specific functions were added to the RingBuilder™ code via conditional cmpilation, to verify that all memory allocated was

de-allocated when the contents of the memory were no longer required. This verification is extremely important because a large amount of temporary data is created during the course of a RingBuilder™ run. A small memory leak can exhaust all available virtual memory on a workstation during a large design run, especially a multi-technology design.

A.2.3 Manual Verification

To ensure that the software was working properly, the design output was manually verified. As a practical necessity, a small test network/demand set was used - one small enough that a person could duplicate the RingBuilder™ functionality using spreadsheets.

A series of 22 test cases were devised to exercise all of the main modules of RingBuilder™. Three main execution paths exist in the code, one for each of the demand route loaders, as shown in Figure 89. Tests were developed to explicitly verify each of these paths.

A small test network consisting of 6 nodes and 8 spans was created and used to verify the correct operation of the code. Manual verification was facilitated by having an independent tester generate a Microsoft Excel spreadsheet for each test case. This spreadsheet was developed in conjunction with Dave Morley from TRLabs Network Systems. A portion of a representative manual verification spreadsheet is shown in Figure 90. It contains network span distance information, cycle circumference information, demand route information, including a summary of demand route assignments on a per-cycle per iteration basis, as well as cycle span utilization information on a per cycle per iteration basis. The verifier manually 'loads' each cycle using the demand route sorting and ring loading heuristics of RingBuilder™. Then balance, capture, and specific progress are calculated. The verifier selects the cycle with the highest figure of merit, notes it as a ring in the final design and manually updates the demand route information by removing the loaded demand route segments from further consideration. The cycles are all re-evaluated in the next iteration. After all demand routes have been loaded, the design is complete. This spreadsheet information is then correlated with the ring selections and the ring loading

results from the corresponding RingBuilder™ design.

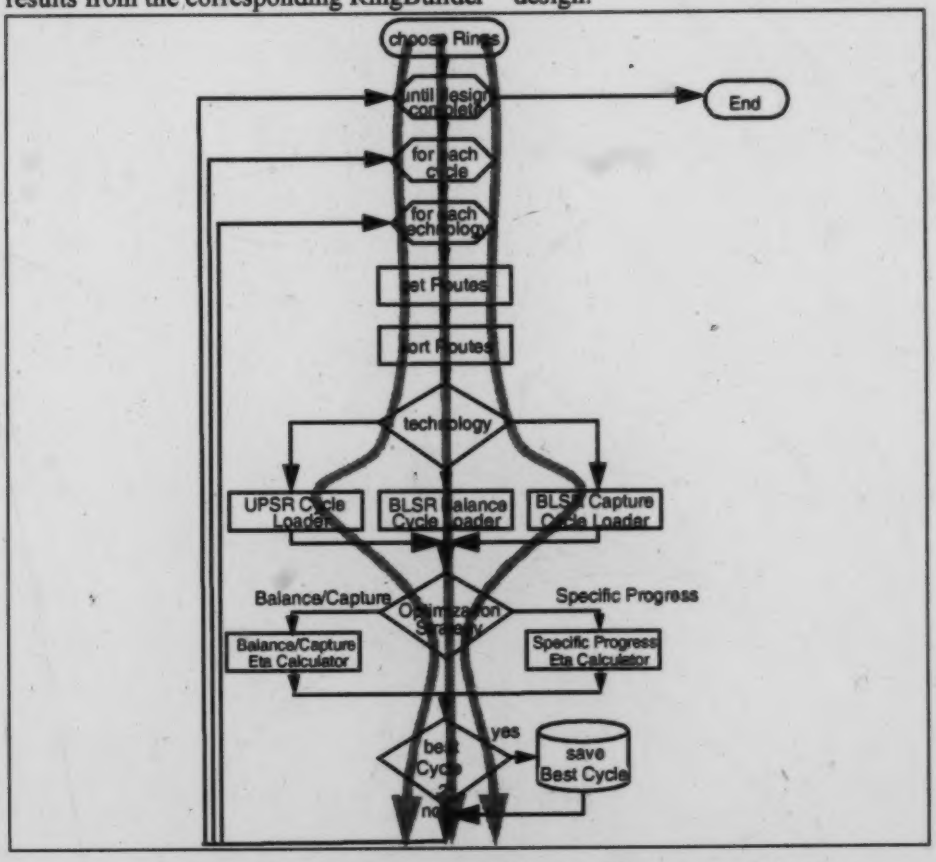
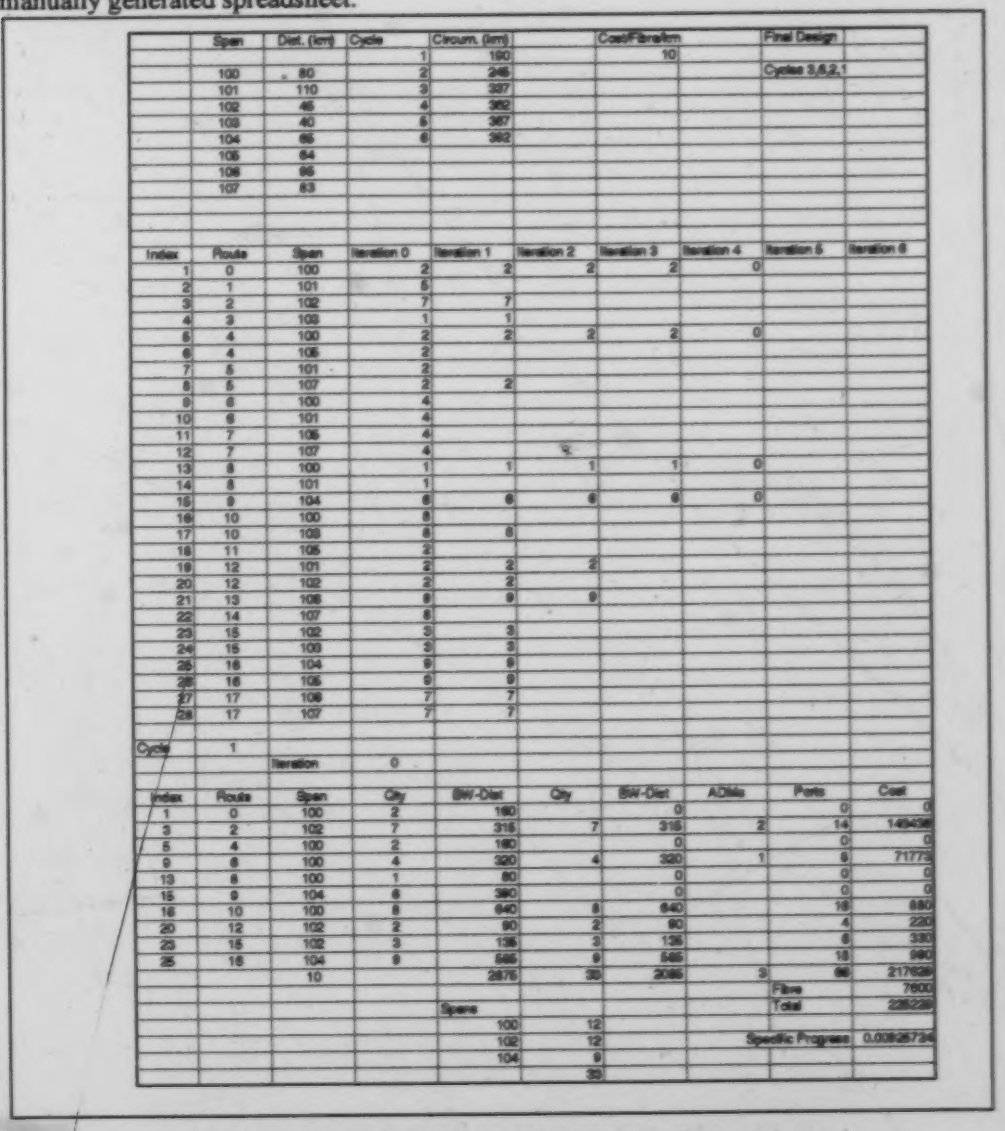


Figure 89: High Level RingBuilder™ Functional Flow

All manually worked test cases were completed by testers who had no previous experience with the iterative synthesis technique used in RingBuilder™. After a verbal briefing covering the heuristic techniques and general methodology of RingBuilder™, the verification team manually produced detailed test results which were compared to a reference suite of the 22 test cases as generated by RingBuilder™. Each time a difference between the manual and the RingBuilder™ results was found, the detailed per cycle per iteration RingBuilder™ output (enabled via conditional compilation) was compared to the

manually generated spreadsheet.



	Span	Dist. (km)	Cycle	Circum. (km)		Cost/Fibre/km		Final Design	
			1	160		10			
	100	80	2	245				Cycles 3,6,2,1	
	101	110	3	337					
	102	45	4	362					
	103	40	5	367					
	104	65	6	362					
	105	64							
	106	66							
	107	63							

Index	Route	Span	Iteration 0	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
1	0	100	2	2	2	2	0		
2	1	101	5						
3	2	102	7	7					
4	3	103	1	1					
5	4	100	2	2	2	2	0		
6	4	105	2						
7	5	101	2						
8	5	107	2	2					
9	6	100	4						
10	6	101	4						
11	7	105	4						
12	7	107	4						
13	8	100	1	1	1	1	0		
14	8	101	1						
15	9	104	6	6	6	6	0		
16	10	100	8						
17	10	103	8	8					
18	11	105	2						
19	12	101	2	2	2				
20	12	102	2	2					
21	13	106	9	9	9				
22	14	107	8						
23	15	102	3	3					
24	15	103	3	3					
25	16	104	9	9					
26	16	105	9	9					
27	17	106	7	7					
28	17	107	7	7					

Cycle	1								
		Iteration	0						
Index	Route	Span	City	BW-Dist	City	BW-Dist	ADMs	Ports	Cost
1	0	100	2	160		0		0	0
3	2	102	7	315	7	315	2	14	149436
5	4	100	2	160		0		0	0
9	6	100	4	320	4	320	1	8	71773
13	8	100	1	80		0		0	0
15	9	104	6	390		0		0	0
16	10	100	8	640	8	640		16	880
20	12	102	2	90	2	90		4	220
23	15	102	3	135	3	135		6	330
25	16	104	9	585	9	585		18	990
		10		2875	33	2085	3	66	217639
								Fibre	7600
								Total	228239
				Spans					
				100	12				
				102	12			Specific Progress	0.03926734
				104	9				
					33				

Figure 90: Sample of Manual Verification Spreadsheet

If a problem was found with the spreadsheet, it was corrected. If a problem was found with RingBuilder™, the code was inspected carefully, modified if appropriate, and re-tested. The 22 test cases were regenerated and compared to the spreadsheet results after every code change. In the development phase of RingBuilder™, 16 of 22 test cases failed initially, with an average of < 10 lines of code affected per test failure. On average only a single fault was found per affected module per failed test case. This indicates that the test

cases were appropriate, as they captured specific functional anomalies quickly.

A.2.4 Implementation Test Network

The 22 test cases exploit 3 different network constraints in 2 categories: node constraints and span constraints. Various system technology combinations were utilized, using both UPSR and BLSR rings with different ADM constraints (port limits and maximum ADM count). Several design methodologies were examined, using different design technology choices (single vs. multi technology), different optimization strategies (balance-capture and direct cost) and different demand route loading strategies (capture biased and balance biased). Only the 'simple' pre-loading demand sorter was used in the test suite.

All test cases used netZb, shown in Figure 91, as the test network.

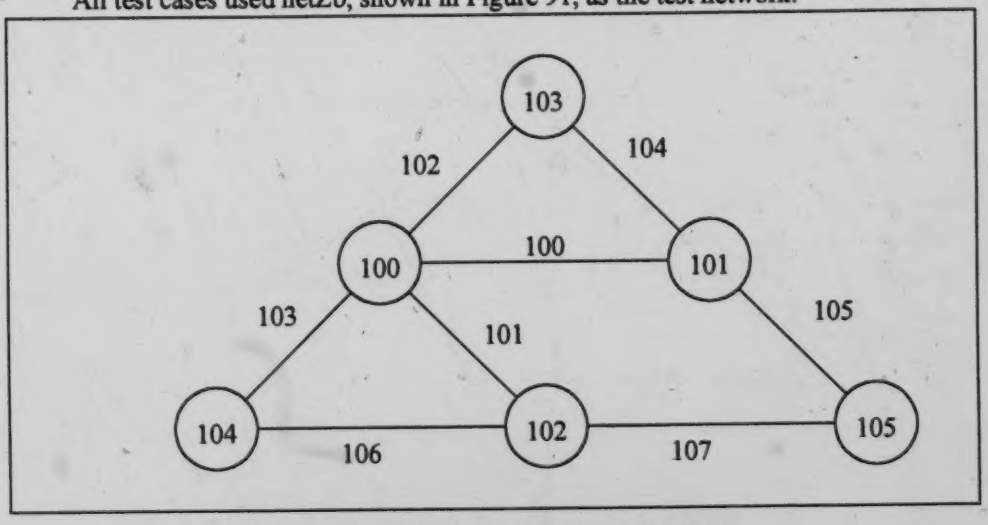


Figure 91: netZb: Test Network for Tests

The point to point demand file for netZb is shown in Table 36.

Table 36: netZb Point-to-Point Demand Information

Point-to-point Demand Number	NodeA	NodeB	Demand Size
100	100	101	2
101	100	102	5
102	100	103	7
103	100	104	1
104	100	105	4
105	101	102	9
106	101	103	6
107	101	104	8
108	101	105	2
109	102	103	2
110	102	104	9
111	102	105	8
112	103	104	3
113	103	105	9
114	104	105	7

The corresponding shortest hop count demand routes are graphically represented in Figure 92.

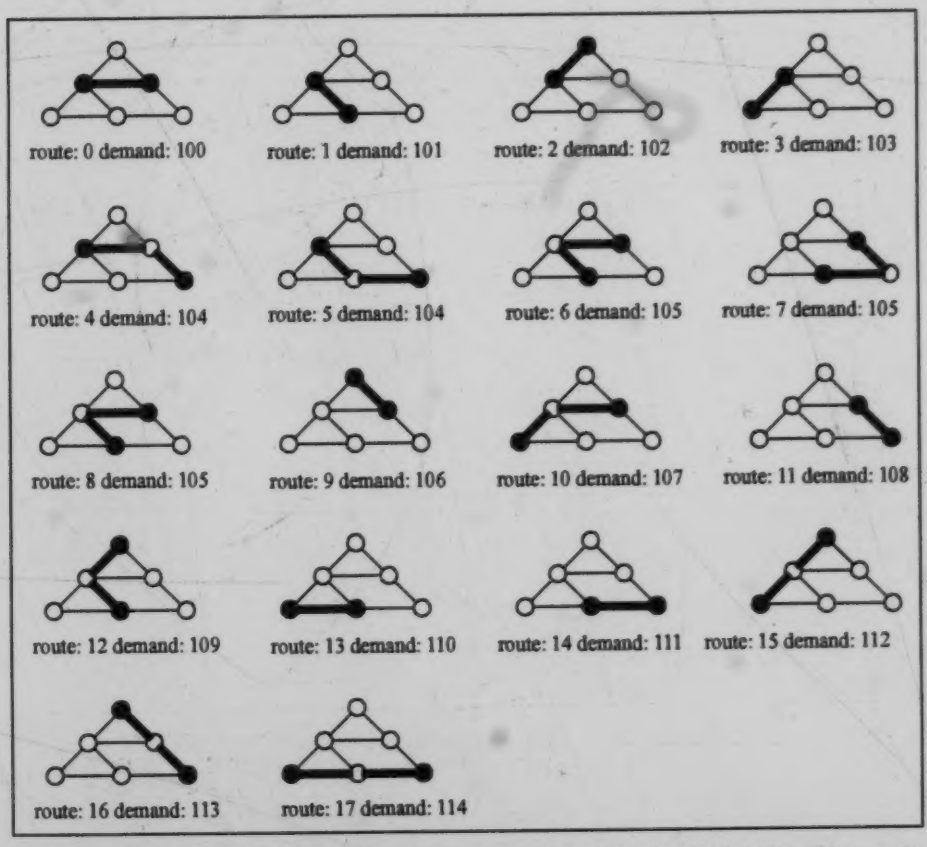

route: 0 demand: 100
route: 1 demand: 101
route: 2 demand: 102
route: 3 demand: 103
route: 4 demand: 104
route: 5 demand: 104
route: 6 demand: 105
route: 7 demand: 105
route: 8 demand: 105
route: 9 demand: 106
route: 10 demand: 107
route: 11 demand: 108
route: 12 demand: 109
route: 13 demand: 110
route: 14 demand: 111
route: 15 demand: 112
route: 16 demand: 113
route: 17 demand: 114

Figure 92: Shortest Path Routes for all Demand Pairs in Validation Network

The cycle set for netZb is shown in Figure 93.

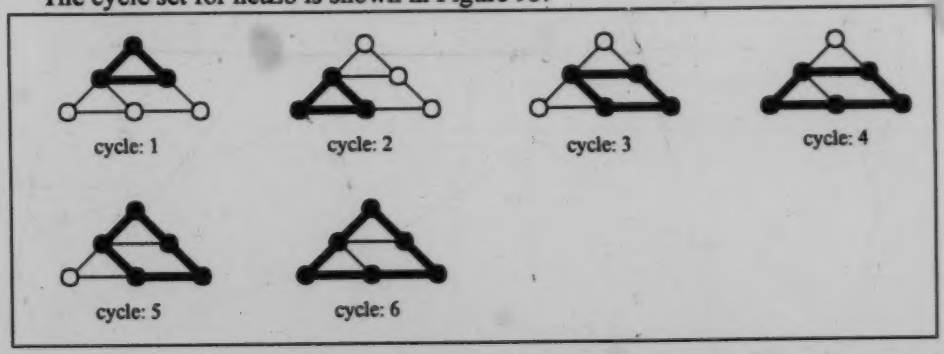


Figure 93: Cycle Set for Validation Network

The span cross sections resulting from k-way shortest path routing of the demand information is shown in Table 37. Figure 92 shows that demand 104 is split into 2 equal-

length demand routes and that demand 105 is split into 3 equal-length demand routes.

Table 37: netZb Span Informtion (with k-way Routing of Demands Span Cross-sections)

Span	NodeA	NdeB	Distance	Working	Spare
100	100	10	80.0	17	0
101	100	10	110.0	14	0
102	100	103	45.0	12	0
103	100	104	40.0	12	0
104	101	103	65.0	15	0
105	101	105	64.0	17	0
106	102	104	95.0	16	0
107	102	105	83.0	21	0

A.3 Test Cases

The test cases exercise funconality that is network specific, ring system technology specific, and design specific. The are outlined in Table 38, which shows how each ring loader is exercised under each costraint class. Only 2 different system types were used: 2-fibre UPSR OC-12 and 4-fibre BLSR OC-12. Only one test case (case8) used multi-technology optimization, producig a heterogeneous SHR design. The other runs used single technology optimization and tus resulted in homogeneous network designs. Direct cost optimization and balance vs. capture optimization strategies were tested. A Node add/drop constraint which disallowed adding or dropping demands at node 100 was applied in test case5, case11, and case15. A node transit constraint which disallowed transiting of demands between rings at node 12 was applied in case6, case12, and case16. An ADM port count constraint which limitd each ADM used to 6 add/drop/transit ports rather than the expected 12 was applied in case3, case10 and case18. An active ADM count constraint which limited the maximum number of ADMs to 2 in any ring deployed was applied in case9, case14, and case19. A span constraint which limited the number of fibres on span 100 to 2 in case13, and to 4 in case7 and case17.

Table 38: Summary of Software Tests

Technology Options	Optimization Strategy	Constraint	Ring Loader		
			BLSR Balance	BLSR Capture	UPSR Capture
Single Technology (4BLSR12 or 2UPSR12)	Direct Cost Optimization	NONE	case1	case2	case4
		Node Add/ Drop	case5	case15	case11
		Node Transit	case6	case16	case12
		ADM Port Count	case3	case18	case10
		Active ADM Count	case9	case19	case14
		Span Fibre Limit	case7	case17	case13
Single Technology (4BLSR12 or 2UPSR12)	Balance/ Capture Optimization	NONE	case20	case21	case22
Multiple Technology (4BLSR12/ 2UPSR12)	Direct Cost Optimization	NONE	case8		case8

Table 39 shows the explicit RingBuilder™ options at the network, system technology, and design level used in each test case.

Table 39: Software Tests by Network, System Technology and Design Considerations

Test Case	Network Considerations					System Type	Design Considerations					
	Network Constraints			ADM Constraints			Design Technology		Optimization Strategy		Candidate Ring Loading	
	Node Constraints		Span Constraint									
	Add/Drop	Transition	Fibre Count	ADM Port Count	Active ADM Limit		Single Technology	Multiple Technology	balance vs. capture	Direct Cost	Capture-Biased	Balance-Biased
case1						4BLSR12	✔			✔		✔
case2						4BLSR12	✔			✔	✔	
case3				12		4BLSR12	✔			✔		✔
case4						2UPSR12	✔			✔	✔	
case5	✔					4BLSR12	✔			✔		✔
case6		✔				4BLSR12	✔			✔		✔
case7			✔			4BLSR12	✔			✔		✔
case8						2UPSR12/4BLSR12		✔		✔	✔UPSR	✔BLSR
case9					2	4BLSR12	✔			✔		✔
case10				6		2UPSR12	✔			✔	✔	
case11	✔					2UPSR12	✔			✔	✔	
case12		✔				2UPSR12	✔			✔	✔	
case13			✔			2UPSR12	✔			✔	✔	
case14					2	2UPSR12	✔			✔	✔	
case15	✔					4BLSR12	✔			✔	✔	
case16		✔				4BLSR12	✔			✔	✔	
case17			✔			4BLSR12	✔			✔	✔	
case18				12		4BLSR12	✔			✔	✔	
case19					2	4BLSR12	✔			✔	✔	
case20						4BLSR12	✔		✔			✔
case21						4BLSR12	✔		✔		✔	
case22						2UPSR12	✔		✔		✔	

A.3.1 Test Case Descriptions

The test cases examine the operation of each of the main RingBuilder™ blocks. Only one parameter is varied in each test case, to facilitate troubleshooting and to simplify manual verification of the result. The test cases are summarized below.

Case1 is the baseline test case. Direct cost optimization is used and the balance-biased route loader is invoked. **Case2** verifies the operation of the capture-biased route loader. **Case3** verifies the node specific ADM port constraint code used in all of the route loaders. **Case4** is the baseline UPSR test case. The UPSR route loader code is tested here. **Case5** verifies the add/drop constraint testing portion of the node constraint code used by the route loaders. Because demands source and terminate at the constrained node (100), the design run terminates without accommodating all point to point demands in the ring

network design. **Case6** exercises the transition constraint testing portion of the node constraint processing code of the route loaders. The design generated accommodates all demands, but no inter-system demand route transiting occurs at node 102. **Case7** exercises the span constraint processing functionality in the route loaders. It limits the fibre count on span 100 of the network to 4 fibres (one system). When the fibre is exhausted on a span, no further systems are routed over that span. The design terminates with some unserved point-to-point demands. **Case8** exercises the direct cost optimization multiple technology comparison capability. It is a two technology design, comparing 2BLSR12 balance routed ring candidates to 2UPSR12 candidates. **Case9** exercises the system level ADM count constraint functionality in the route loaders. This design consists of systems which are equivalent to point-to-point diversely routed systems. **Case10** exercises the ADM port count limit testing code within the UPSR route loader. **Case11** exercises the node add/drop constraint testing code within the UPSR loader. The generated design is incomplete because it does not route any demands that start or terminate at node 100. **Case12** exercises the node transit constraint processing functionality of the UPSR loader The design generated is complete, but there is no inter-ring transit activity at node 102. **Case13** exercises the span constraint code in the UPSR route loader. The design generated is incomplete because only a portion of the demand traversing span 100 is picked up by the one system, and no further systems are allowed to use that span. **Case14** is another 2UPSR12 run, exercising the active ADM count processing code in the UPSR route loader. The number of active ADMs per system is limited to 2 in this design, which can be directly compared to the baseline UPSR design in case4. **Case15** exercises the BLSR capture loader node add/drop constraint processing functionality. The design generated is incomplete as demands that source or sink at node 100 cannot be loaded. **Case16** exercises the BLSR capture router's node transit constraint processing. A complete design is generated, but no transiting between systems at node 102 occurs. Compared to the baseline capture-routed 4BLSR12 design in case2, demand route 5 no longer transits between systems at node 102. **Case17** exercises the span fibre limit processing functionality of the BLSR capture loader. The design generated is incomplete, with routes 0, 4, and 8 not completely loaded onto systems. **Case18** exercises the node constraint processing code in the BLSR capture loader. **Case19** exercises the active ADM count constraint processing of the BLSR

capture loader. The design generated consists of what amounts to a series of point-to-point systems. **Case20** verifies the balance vs. capture eta calculator for the BLSR balance loading scheme. Balance vs. capture optimization is used instead of direct cost optimization via specific progress. The design is generated with a balance bias factor α, of 0.15. **Case21** verifies the balance vs. capture eta calculator for the BLSR capture loading scheme. The design is generated with a balance bias factor α, of 0.85. **Case22** verifies the balance vs. capture eta calculator for the UPSR capture loading scheme. The design is generated with a balance bias factor α, of 0.6.

A.4 Implementation Testing Findings and Observations

The software verification process revealed several significant points. Memory leak testing emphasized the importance of writing the memory allocation code and the memory deallocation code at the same time to prevent problems. Functions written in this manner were much less troublesome to debug. In the cases where memory deallocation code had to be written after the fact, memory leaks were much more likely to occur.

Three points became apparent with respect to the manual verification process. First, most problems were identified in the low level span and node check functions of the ring loaders. Second, the manually generated result and the RingBuilder™ generated result could differ, even though both results were correct. Last, even with 22 manually generated test cases, some software bugs were not revealed until larger network designs were tackled.

The majority of software bugs were identified in the span and node examination functions because that is where most of the constraint processing algorithms exist. The constraint processing algorithms are the heart of RingBuilder™'s capability to generate 'real-world' realizable designs. These functions are the most utilized code in the RingBuilder™ framework. The forward- and reverse-span/node check functions and the terminal-span/node check function are used once per hop per demand per cycle per iteration. As a result, even subtle problems with this code were quickly identified.

Often discrepancies appeared between the manually generated reference result and the result generated by RingBuilder™ even though both were correct. These differences

appeared when different choices of equal value were made in the route sorting process prior to route loading. If a tie occurred between 2 or more routes, the human verifier and RingBuilder™ sometimes made different choices, resulting occasionally in quite dissimilar designs. The solution from a verification procedure perspective was to generate a second manual verification, and to follow the choices made by RingBuilder™ (quicksort routine) under the 'tie' conditions. This is not a good answer, however, because the quicksort routine implementation varies by computing platform, and RingBuilder™ was run on multiple computing platforms in the course of this work. A better solution would have been to generate a 'RingBuilder™QuickSort', but since that routine would only have facilitated manual verification, it was deemed unnecessary.

While the manual verification process was a practical to debug a substantial amount of code, several coding problems were discovered in the course of generating the experimental results of Chapter 5. These new problems tended to be quite subtle and only manifested themselves in the course of generating results on large network designs. The problems were always discovered through the detailed examination of the output files. Troubleshooting was carried out via the conditionally generated diagnostics developed for problem isolation and correction in the software verification process. Without these diagnostics, it would have been impossible to isolate the problems because the network designs were too complex to generate manually using RingBuilder™ heuristics.

Overall, the software testing process was valuable in the development of sound code, but surprisingly, not all problems were identified until more realistic network problems were tackled.

A.5 Conclusion

This Appendix described the 22 test suites which were used to verify the functionality of the RingBuilder™ framework through the comparison of RingBuilder™ output with manually generated solutions for each of the cases. These tests, combined with memory leak testing, ensure that networks designed with RingBuilder™ will be realizable. All of the functional blocks exercised in this Appendix were used in the generation of the real-world network designs in Chapter 5.

B. RingBuilder™ Direct Cost Optimization via Specific Progress Example

This appendix illustrates the operation of the direct cost optimization via specific progress technique used by RingBuilder™. The ring cost components considered for costing purposes is illustrated in Figure 94.

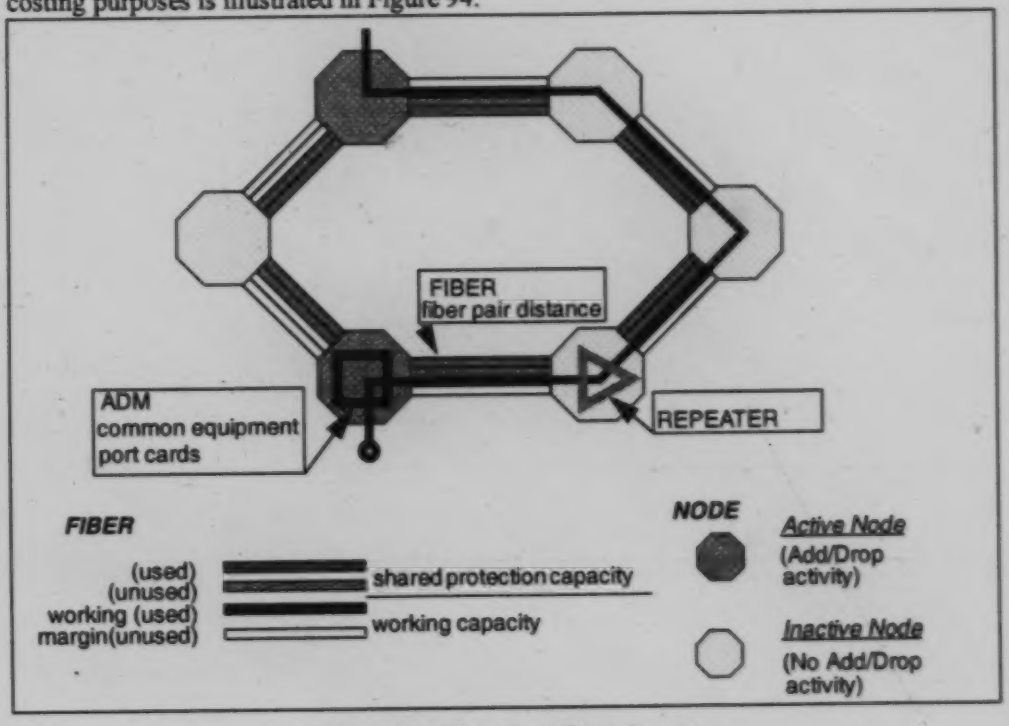


Figure 94: Ring Cost Components

The measurement of design progress for a ring carrying demand segments is illustrated in Figure 95.

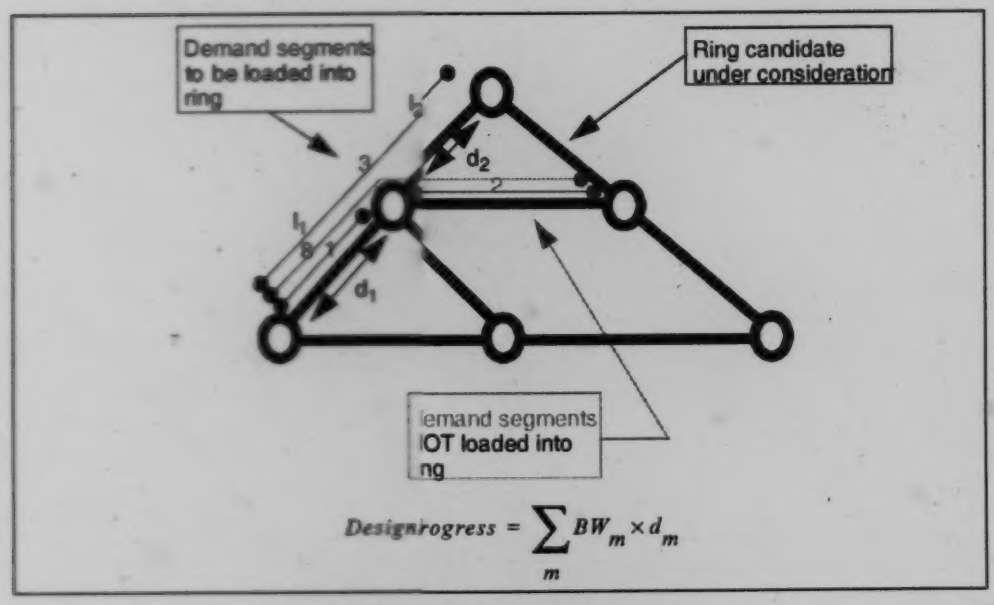

Demand segments to be loaded into ring
Ring candidate under consideration
Demand segments NOT loaded into ring
$Design_{Progress} = \sum_{m} BW_m \times d_m$

Figure 95: Ring Progress

The costing process for a ring candidate is illustrated in Figure 96.

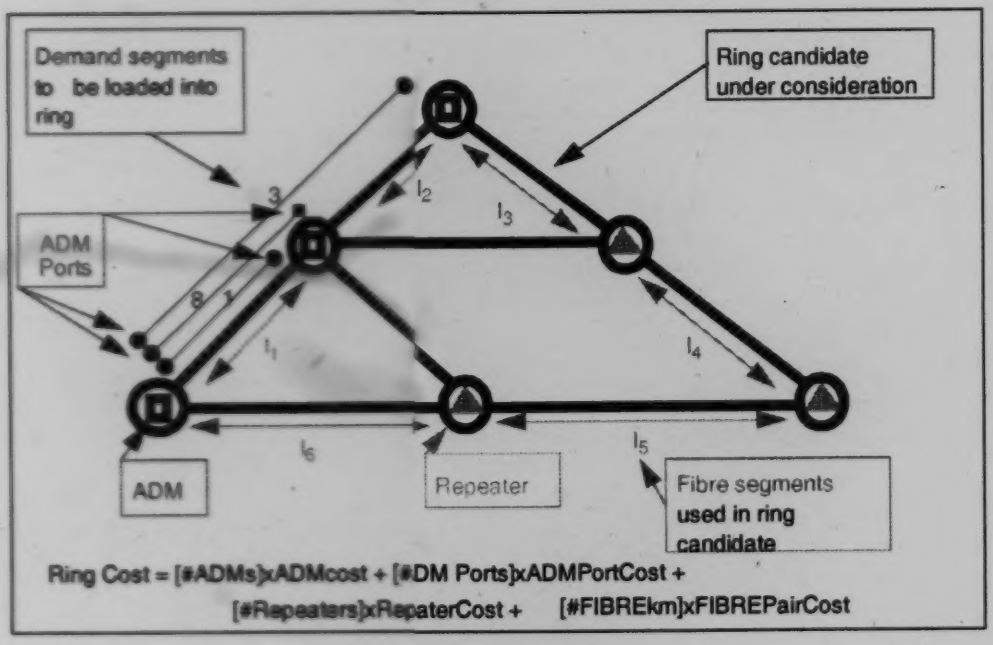

Demand segments to be loaded into ring
Ring candidate under consideration
ADM Ports
ADM
Repeater
Fibre segments used in ring candidate
Ring Cost = [#ADMs]xADMcost + [#DM Ports]xADMPortCost +
[#Repeaters]xRepaterCost + [#FIBREkm]xFIBREPairCost

Figure 96: Ring Costing

The cost factors for the design example are given in Figure 40.

Table 40: Design Example Cost Factors

ITEM	OC-12	OC-48	Comment
ADM	0.05	0.1	
ADM Port Card	0.0025	0.005	
Repeater	0.025	0.05	
Fiber	0.004	0.004	
DCS	1.0		1. Other item costs are normalized to DCS cost. 2.Used in actual design costing -- not used in individual ring selection
DCS Port Card	0.02		

The routed demands, the network and the cycle set for the network are illustrated in Figure 97. All of the demands have been shortest logical path routed. Two technologies are considered in this design example, 2BLSR12 and 2BLSR48. The design proceeds through 4 iterations, with each cycle being considered with each technology in each iteration. The design terminates when all unserved demand has been loaded onto rings. The design iterations are shown in Figures 98 through 101.

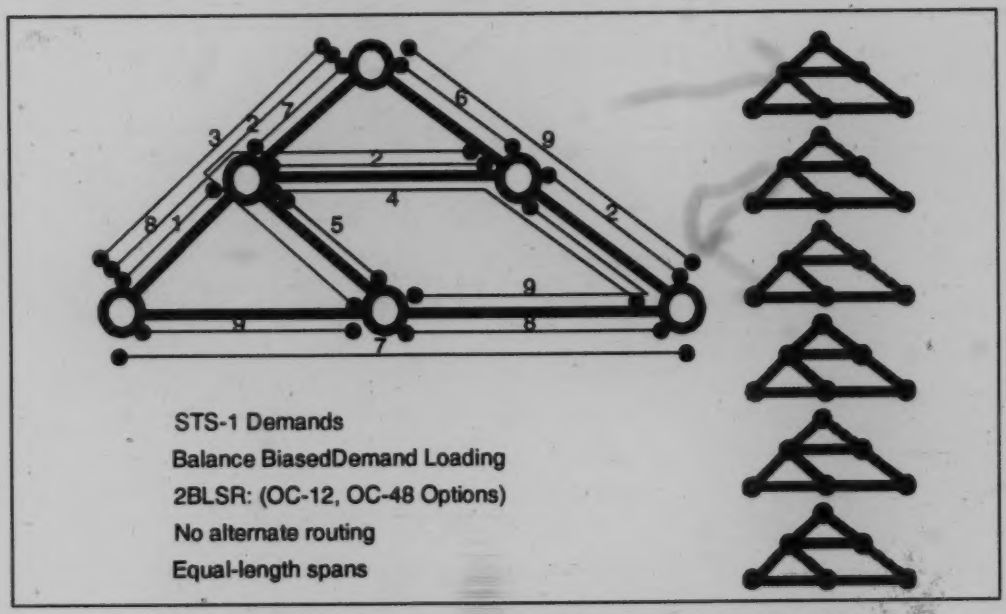


Figure 97: Design Example: Routed Demands and Cycle Set

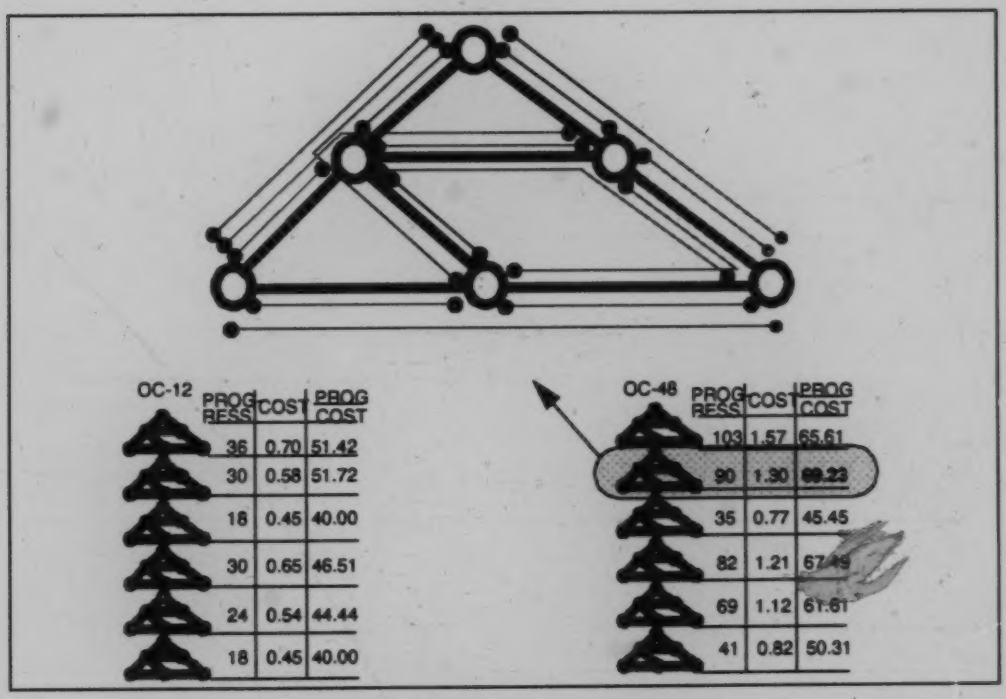


Figure 98: Design Example: Iteration 1

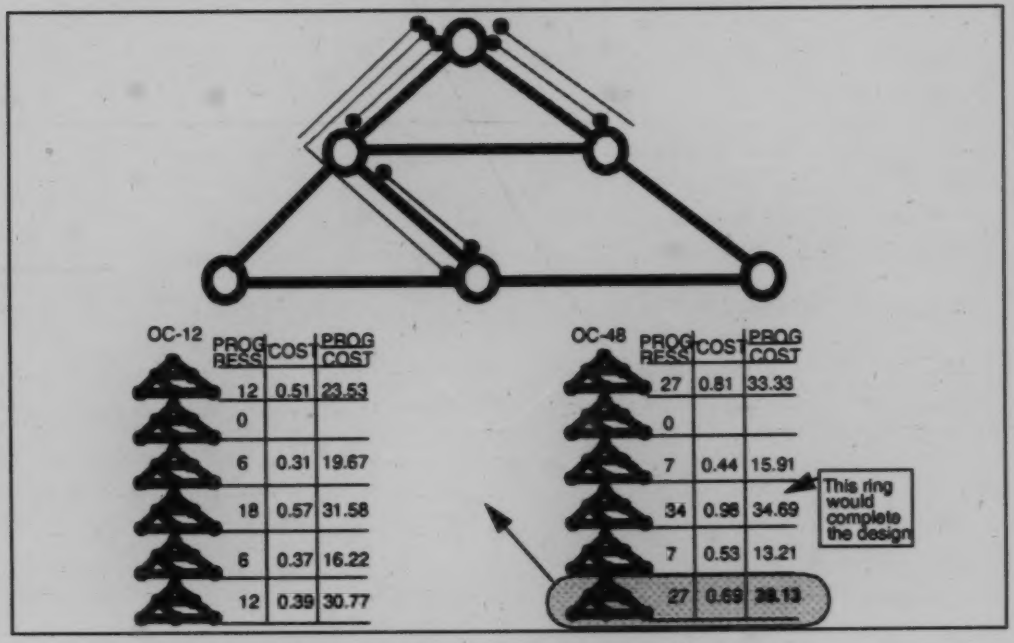


Figure 99: Design Example: Iteration 2

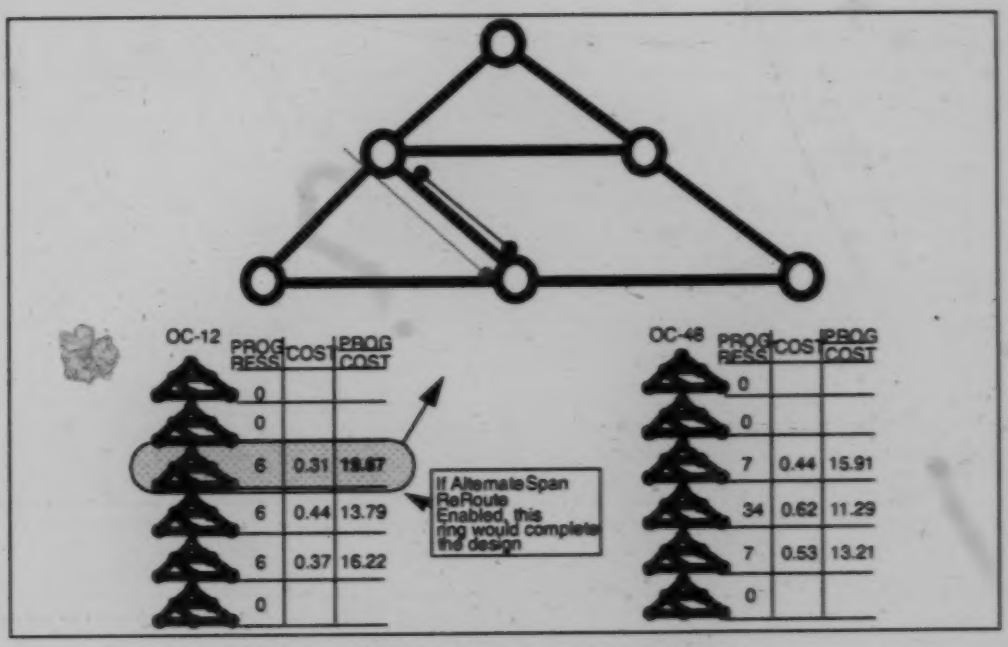


Figure 100: Design Example: Iteration 3

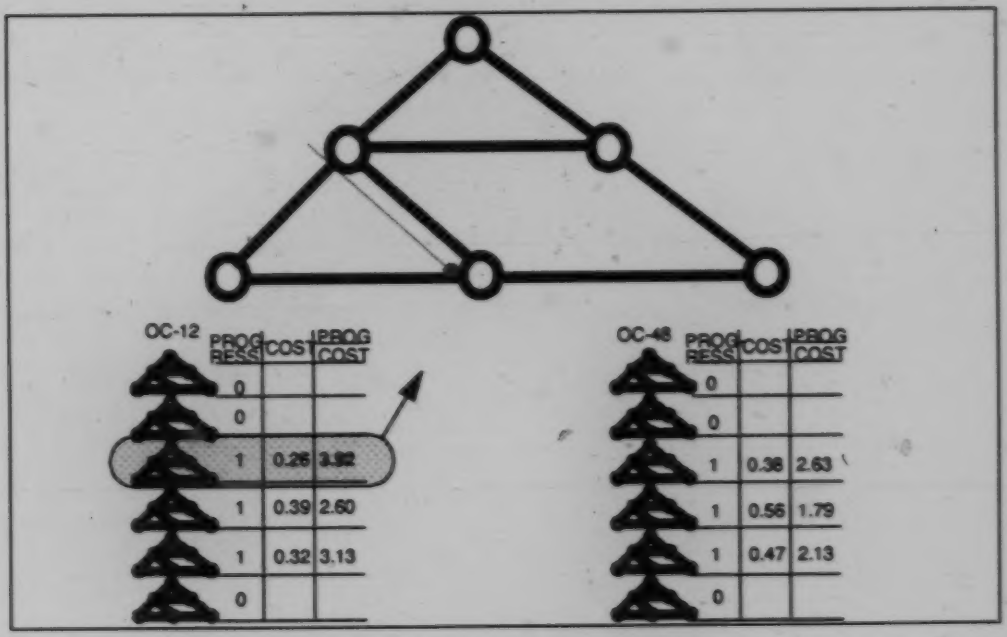


Figure 101: Design Example: Iteration 4

C. RingBuilder™ netJ Baseline (Case23) and netM Baseline (Case26) Ring Designs

This appendix pictorially describes the baseline ring designs for netJ and netM. Nodes with ADMs are shown as ● , and glass-throughs are shown as o .

C.1 netJ (Metropolitan Network) Baseline: Case23

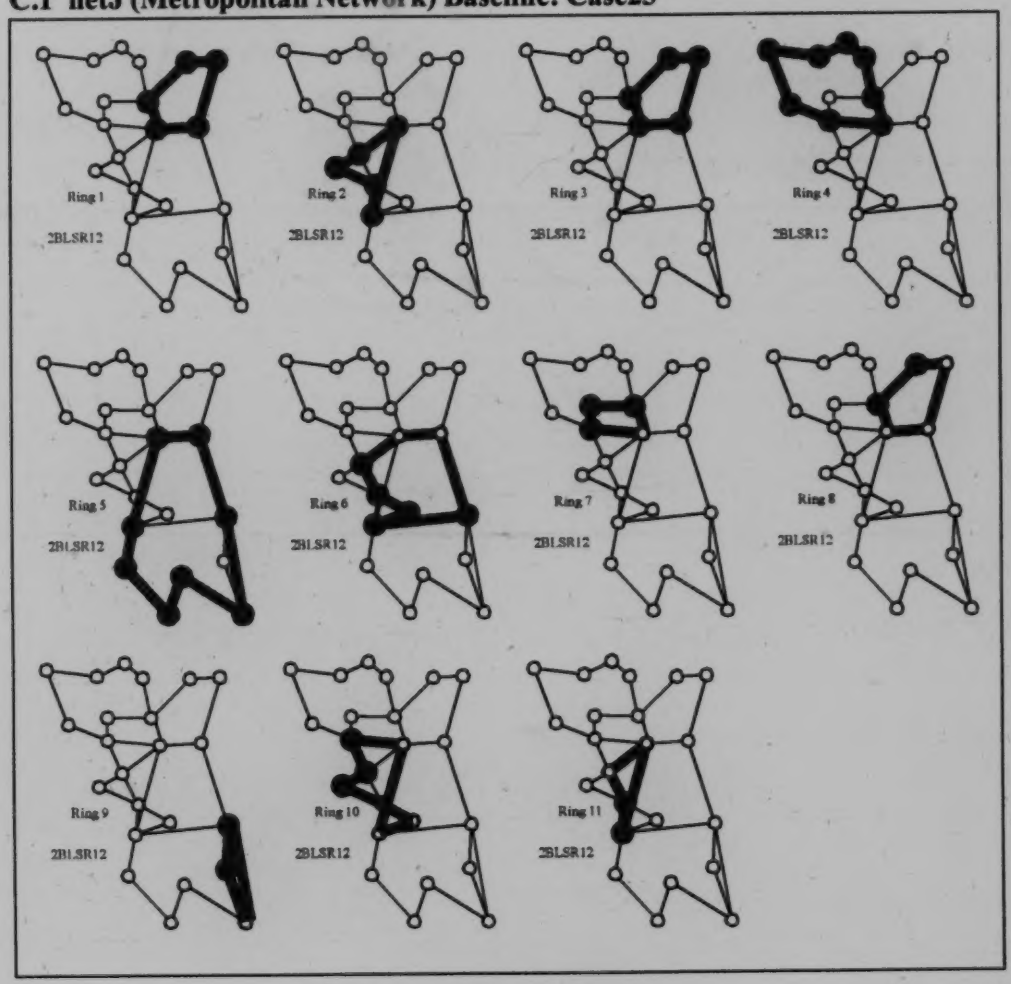


Figure 102: Case23 Design

C.2 netM (Long-Haul Network) Baseline: Case26

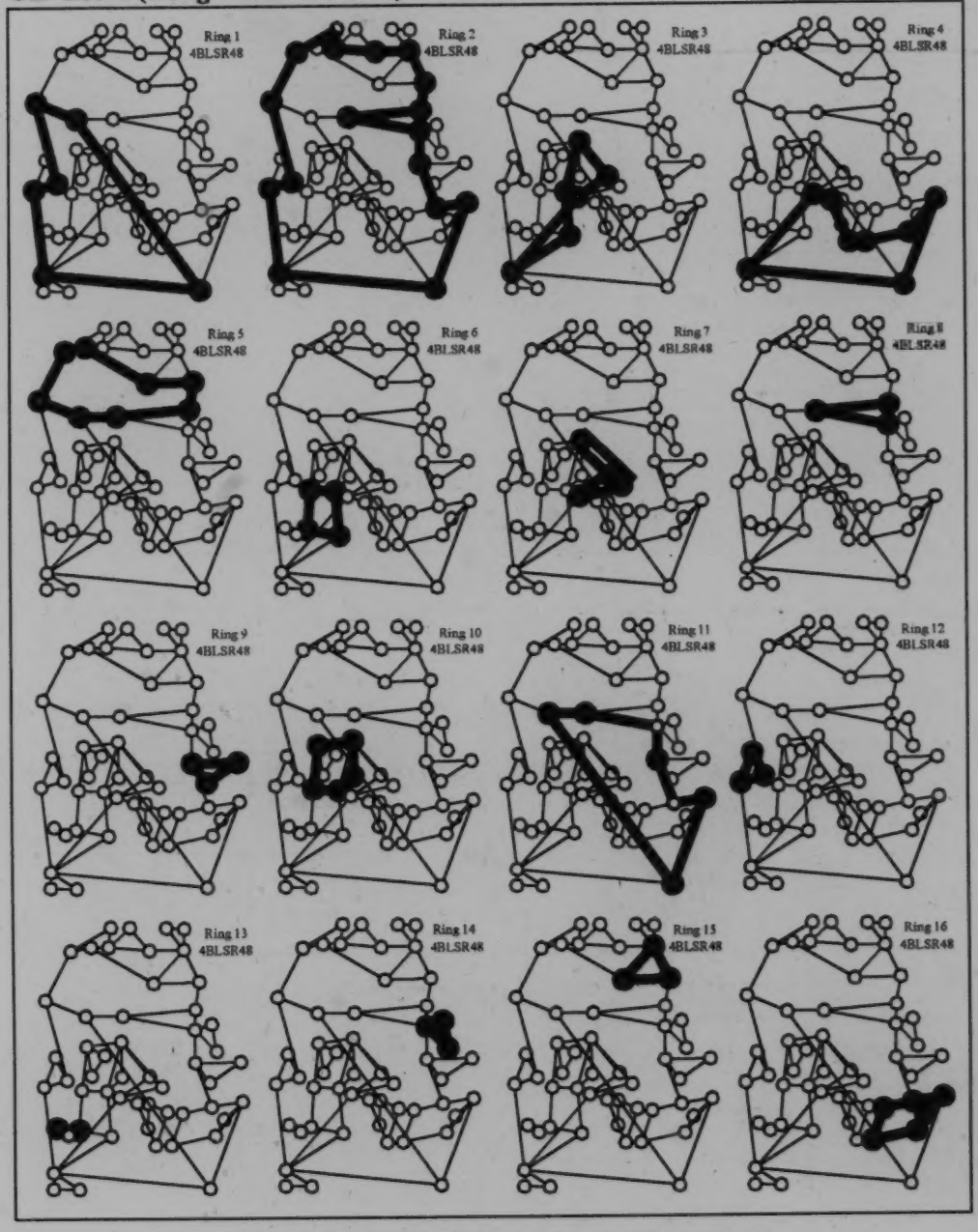


Figure 103: Case26 Design -- Rings 1 Through 16

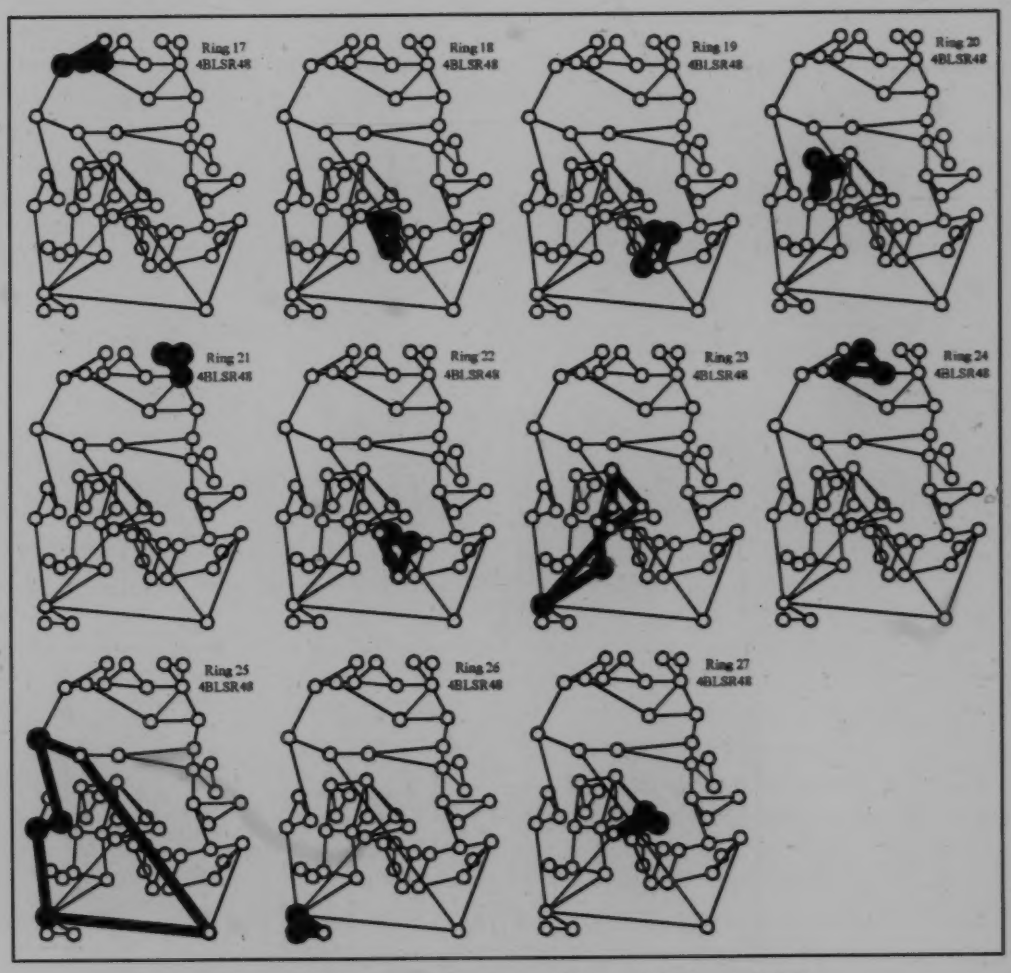


Figure 104: Case26 Design – Rings 17 Through 27